MISSION PLANNING FOR CLOSE-PROXIMITY SATELLITES

THESIS

Barry R. Witt, Captain, USAF
AFIT/GA/ENY/09-M10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GA/ENY/09-M10

MISSION PLANNING FOR CLOSE-PROXIMITY SATELLITIES

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Barry R. Witt, BS

Captain, USAF

March 2009

AFIT/GA/ENY/09-M10

MISSION PLANNING FOR CLOSE-PROXIMITY SATELLITES

Barry R. Witt, BS
Captain, USAF

Approved:

/signed	
William E. Wiesel (Chairman)	Date
/signed	
Kerry D. Hicks (Member)	Date
/signed	
Douglas D. Decker (Member)	Date

Abstract

Given an initial orbit and a set of other orbits of interest, the path requiring the lowest change in velocity between them is of high value. Software was developed to visualize the change in velocity required from a user defined window of burn to a user defined window of intercept. The time between burn and intercept is unrestricted. Multiple intercepts require searching an n! solution space. Research then is focused on ways to pick optimal choices without fully calculating out the cost of doing all possible combinations. Some code was translated from BASIC and the rest was newly developed in MATLAB. The code was validated using orbits with known intercept solutions.

Soli Deo Gloria

Acknowledgements

I would like to express my sincere appreciation to my faculty advisor, Dr William Wiesel, for his guidance, support and computer code throughout the course of this thesis effort. The insight, experience and extension on the p-iteration technique was certainly appreciated and in the case of the code, necessary for this thesis. I would, also, like to thank the help of Dr David Vallado and Dr Jeff Beck with using their computer code. Their pioneering work using the SGP4 propagator was fundamental to the success of the code presented here.

Barry R. Witt

Table of Contents

List of Figures

Page

List of Tables

Page

MISSION PLANNING FOR CLOSE-PROXIMITY SATELLITIES

I. Introduction

Close-proximity operations have long been of interest to space-faring nations. The ability to observe and remotely sense other objects in orbit is of great interest to the Department of Defense (DoD). Since the beginning of the space-age, obtaining a clear understanding of the state of your space vehicle as well as the state of others has been relatively difficult. Onboard sensors and ground-based telescopes have only allowed a small fraction of the total state of the satellite to be known. Current technology allows some information to be gained from a distance. Radar provides imagery, detection and measurement data about the orbit a satellite is in and optical telescopes (usually incorporating adaptive optics) can provide visual images for dimensional analysis. Various types of antennae can be used to pickup electromagnetic radiation emanating from the satellite of interest. The knowledge of the orbits and operations of satellites can be described as having space situational awareness (SSA).

Recent missions in close-proximity satellite operations have allowed a better understanding of the operations of other satellites and thus have increased our SSA. Missions such as XSS-10 (Figure I-1) and XSS-11 (Figure I-2) advanced the state of technology in this area. XSS-10 (Air Force Research Labratory, 2005) was a simple proximity mission around the upper stage it was boosted into orbit on (Figure I-3). XSS-11 (Air Force Research Labratory, 2005) demonstrated the ability to change orbits and intercept other non-related objects. Mission planning for these and future operations are thus of high importance to the United States Air Force.

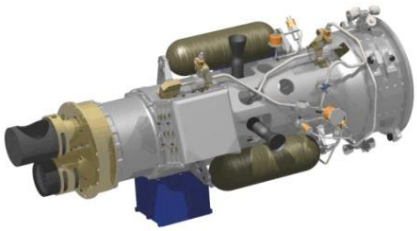


Figure I-1. XSS-10

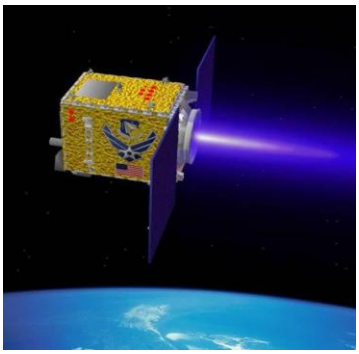


Figure I-2. XSS-11

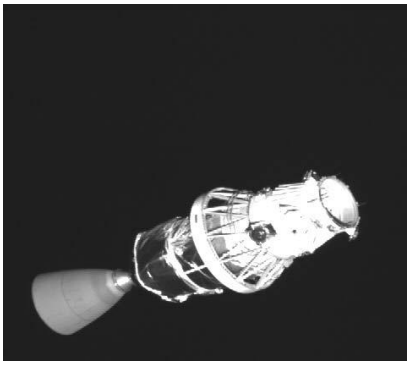


Figure I-3. Picture of upper stage from XSS-10

The general class of missions undertaken by close-proximity satellites is known as intercept missions. These are characterized as matching the position vectors of the interceptor and target but in general not their velocity vectors. Although there are different classes of engines available for on-orbit operations (chemical, electrical, nuclear, etc.), the research conducted here focused on an interceptor

powered by a conventional chemical engine. Thus, an instantaneous change in velocity without a change in position is the chosen method of acceleration. Depending on what is valued, there are a myriad of ways to approach this type of mission. As refueling a spacecraft in orbit is still under development, usually finding low change in velocity values (delta V) is the cost function of choice. In other scenarios, finding low intercept times may be more valued. Chioma (Chioma V. J., 2007) presented different ways of finding the proper burn with developed cost functions. The term "burn" is used to describe the delta V which puts the interceptor into an orbit which will intercept the target. The research conducted here used the two positions and time-of-flight method in finding the appropriate burn. Of note is the expansion of this well known problem with multiple revolutions. In-other-words the interceptor can complete multiple orbits before the intercept occurs. With the developed code one spot for investigation was determining if the lowest delta V occurs when the interception happens at what is known as the relative line of nodes. (This line is constructed by connecting the two points formed by the intersection of the original interceptor orbit and the target orbit.)

Existing BASIC code developed by Dr. Wiesel was translated into MATLAB. Preexisting SGP4 propagation MATLAB code developed by David Vallado and translated by Jeff Beck was used. The rest of the code was developed by the author. The code allows the user to define a window and a time step to use in that window for the burn as well as for the intercept. For each combination of proposed burn and proposed intercept times within those windows the code may return up to two required velocity vectors for interception. Only valid vectors (orbits with a perigee height of at least 200 km) were considered and then the smallest delta V was picked. The delta V's then can be used to fill an array with burn and intercept times (the axes aren't different windows, but different times within the windows.) This can be represented in a surface plot or a 2-D color plot to easily pick out the lowest values. As shown in Section III, patterns emerge and can be used in the analysis of different proposed orbits.

Unfortunately, there are limitations to the research conducted. The user must create burn and intercept windows from which the software finds the delta Vs. Without having a proper understanding of the problem space, the user-defined windows could create conditions where the overall optimal solution would not be found. For example, as will be shown, the optimal solution occurs when the target is along the line of nodes. The user may create burn and intercept windows during which time the target is never on the line of nodes and thus miss the lowest possible delta V for the intercept. This research examined the problem of intercepting more than one satellite in a consecutive manner. For this type of problem (referred to as "multiple burn"), only the six classical orbital elements (COEs) and the epoch year and day are updated in the Two-Line Elements (TLEs.) TLEs capture more information than this about the orbits. This additional information allows a propagator to better model accelerations felt by the spacecraft and thus the updated TLEs used are not as accurate about the new orbit as they were about the original orbit. Finally, only a rudimentary analysis of a small data set was completed. Much more work in this area should be done.

II. Problem Setup

The problem at hand belongs to the orbital mechanical class of problems called interception. A basic intercept is changing the orbit of one satellite (the interceptor) so that it will have the same position vector as another satellite (the target) at a given time. Orbits are governed by Newton's Law of Gravitational Attraction as shown in Equation 1.

$$F = G\frac{m_1 m_2}{r^2} \tag{1}$$

where F is the gravitational force between the two point masses, G is the universal gravitational constant, m_1 is the mass of the first point mass, m_2 is the mass of the second point mass, and r is the distance between the two point masses. A satellite is not a point mass and the Earth is certainly not one either. The forces exerted on a spacecraft by the Earth are approximated by the propagator used in this research.

Once the burn is made the interceptor will intercept the target. It may be required to repeatedly intercept the target. This can be accomplished by changing the orbital period of the interceptor to equal that of the target. In this way the two satellites will periodically have the same position vector (usually once per orbit). This is called a repeated intercept. After the desired number of intercepts has been achieved the interceptor can change its orbit again in order to intercept another target. The work done here will look at solutions for the multiple burn problem. The research does not look at the repeated intercept problem. Although it is physically possible to continuously accelerate in a satellite's orbit, chemical propulsion systems are best used in single burn impulses. When evaluating the burn required for interception, there were no restrictions placed on the ability of the interceptor to complete the burn. That is to say, onboard propulsion systems of a typical interceptor would probably not be able to go from 7 km/s in one direction to 7 km/s in the complete opposite direction, instantaneously. But this study does not eliminate these unrealistic answers. Thus plots showing delta

V solutions for certain burn/intercept combinations with twice nominal orbital velocities are examples of not physically realistic intercept scenarios. Perfect knowledge of the interceptor's orbit and target orbit was assumed. Furthermore, no burn errors were introduced.

Intercept problems have been looked at extensively in the literature. The solution approach taken in this research is the two positions and time-of-flight problem. This problem (sometimes called the Gauss Problem) takes the burn position and the intercept position with the required time of flight and returns the velocity required at burn and the velocity at intercept of the interceptor. One method for solving the Gauss Problem is called the p-iteration technique. As described in Bate, Mueller and White, "the method consists of guessing a trial value of p from which we can computer the other two unknowns, 'a' [the semi-major axis] and delta E. The trial values are checked by solving for t and comparing it with the given time-of-flight." (Bate, Mueller, & White, 1971, p. 241) The "p" described is also known as the semi-latus rectum, and is the line drawn from a foci of an ellipse, perpendicular to the semi-major axis, to the ellipse itself. "E" is the eccentric anomaly. The authors develop a method that finds the slope of the t vs p curve and thus allows the classical Newton Root finder to be used to adjust p. The MATLAB file "piter.m" (translated from BASIC, developed by Wiesel) applies this method with "tofp.m" finding the time-of-flight for the proposed p. Collinear burn and intercept position vectors (defined as the angle between them having a cosine value greater than 0.985 or less then -0.995) were not considered. It is well known in the two position vectors and time-of-flight problem space that collinear position vectors are a singular case. Collinear position vectors move the problem into another type of intercept problem, one of the most recognizable being the Hohmann Transfer case.

The intercept problem also examined the case where the target is at the relative line of nodes. Any two orbit planes with different inclinations will intersect each other along a line. The line drawn through these two points is called the relative line of nodes. Other than a change in altitude, the trajectories of the two satellites intersect and this then is obviously a natural place to plan on an

interception. Changing the altitude of an orbit and phasing the spacecraft in its orbit are relatively cheap maneuvers. Completing an inclination change can be much more expensive as shown in Equation 2. (Titus, 2008)

$$\Delta V = 2V \sin\frac{\Delta i}{2} \tag{2}$$

where V is the magnitude of the initial and final velocities, Delta V is the magnitude of the delta V vector and delta i is the change in inclination. The extraordinary cost of inclination changes is illustrated by an inclination change of 60°, which would result in a change in velocity of magnitude equal to orbital velocity! This high cost leads us to look for a low cost solution elsewhere. It is proposed that the lowest change in velocity required for an interception will occur when the point of interception is on the relative line of nodes.

The propagator of choice for this research is called Simplified General Perturbations No. 4 (SPG4) (Hoots & Roehrich, 1980). Over the years, there have been updates to the model which have allowed it to work properly with released DoD TLE data (Vallado, Crawford, Hujsak, & Kelso, 2006). SPG4 allows data for satellites (in TLE format) to be used to propagate a satellite's orbit over time. Software from Dr Vallado (Vallado D. , Celestrak: Astrodynamics Software, 2009) was used. The MATLAB file "twoline2rv.m" (translated from C by Beck, developed by Vallado) allows TLEs to be read in and initializes the propagator "spg4.m" (translated from C by Beck, developed by Vallado). These files use a small number of other utility files written by Vallado and Beck. As SPG4 is designed for Low Earth Orbit (LEO), only satellites in LEO were examined.

The crux of the problem is calculating the change in velocity. The vector mathematics for finding a delta V is simply the change required in the velocity of the interceptor. This is shown in Figure II-1.

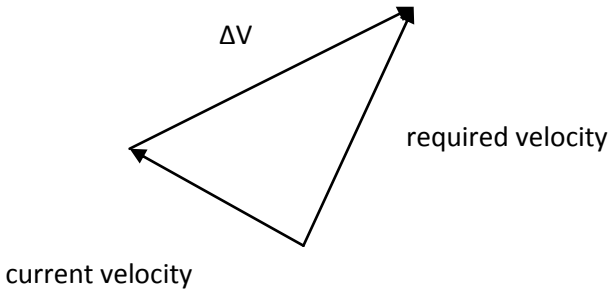


Figure II-1. Vector Geometry of Burn

From Figure II-1 we get Equation 3.

$$\Delta\vec{V} = \vec{V}_r - \vec{V}_C \tag{3}$$

where $\vec{V}_r$ is the required velocity and $\vec{V}_C$ is the current velocity. SPG4 provides the velocity of the interceptor at any time (i.e. the current velocity). The p-iteration technique provides the required velocity vector.

In an effort to be highly automated, the code developed for this research uses a number of nested "for" loops, as illustrated in Figure II-2. For each proposed burn position, a delta V can be found for all of the proposed intercept positions. This is shown by the feedback loop on the left. Once all of the delta Vs have been found for the first burn position, the next burn position is used (the feedback loop on the right) and the first loop begins again. The burn positions come from the proposed burn window used.

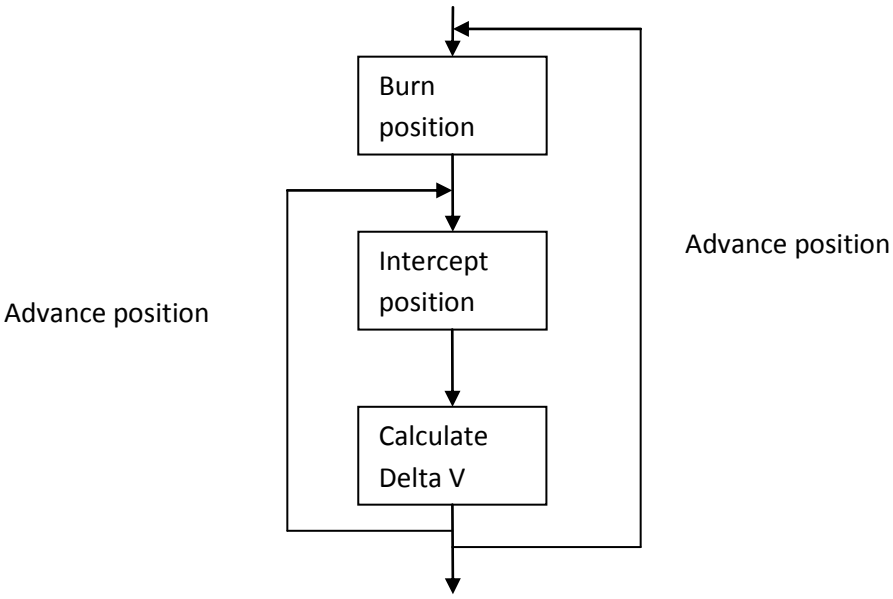


Figure II-2. Method of Calculating Delta V's.

The script then can be run with prompts for the user to guide the process of creating the burn and intercept windows. After calculating the delta V's for all burn and intercept combinations, the user can select a burn/intercept combination or the computer can find the combination with the lowest delta V.

For the code developed, the input variables are: two position vectors, time-of-flight, number of revolutions, and a short-way or long-way switch. The two position vectors are the position at burn and the position at intercept. The time-of-flight is the time requested between burn and intercept. The number of revolutions is not dependent on the time-of-flight; rather, it is an independent variable. For example, to arrive at the intercept position in 6 days, two possible intercept orbits are one that goes around the moon and another that stays in LEO. Thus the first intercept orbit has less than one Earth revolution, while the second has multiple. The research conducted only looked at 5 different proposed number of revolutions for each burn/intercept combination. The number of revolutions search was centered on the number of revolutions the interceptor would take for the time-of-flight used if its orbital period did not change. Two revolutions below through two above the nominal number of

revolutions were used in the calculations. The last variable determines which way the direction of travel will be for the proposed interception orbit. This is shown in Figure II-3 (as adapted from (Bate, Mueller, & White, 1971) Figure 5.2-1).

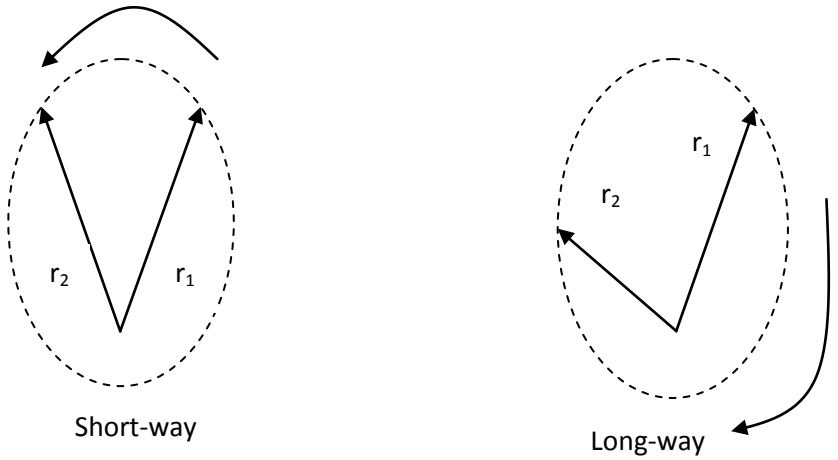


Figure II-3. P-iter Geometry

The code discounts orbits that have a perigee height of less than 200 km above the surface of the Earth, as well as negative time-of-flight (when the burn and intercept windows overlap) and negative proposed number of revolutions. The perigee height check is used to prevent using orbits that go through the Earth (having a perigee height less than the radius of the Earth). A perigee altitude of 200 km was used so that valid orbits would not have the interceptor degrading soon after the burn. The code then selects the smallest delta V value of the valid velocity vectors. As shown in Figure II-4 there could be up to 40 vectors to evaluate for each burn and intercept combination. When this has been completed for all combinations, the arrays are plotted.

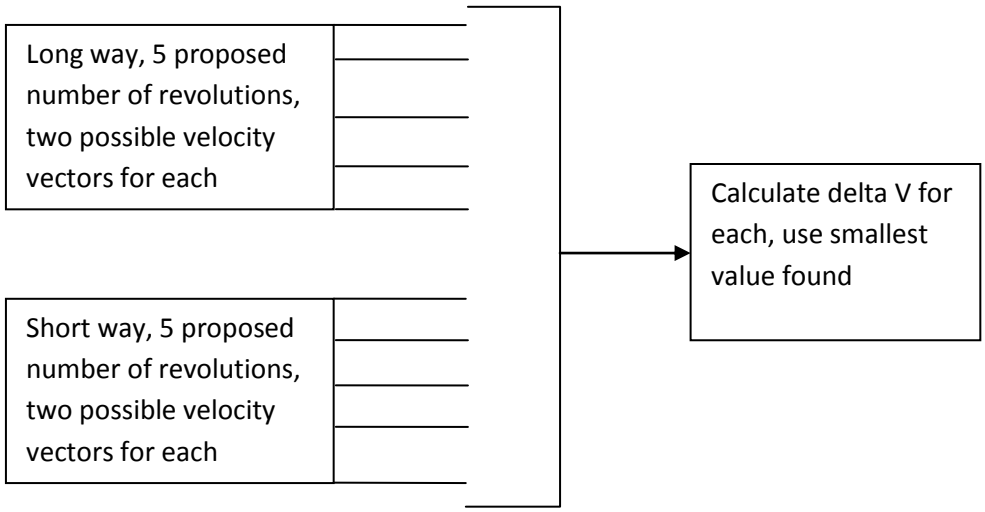


Figure II-4. Total Calculation for Each Burn/Intercept Combination

The “single burn” case is when only one satellite is being examined for interception. The “multiple burn” case is when after intercepting one satellite, the same burn/intercept analysis is run on the next satellite using the new orbit the interceptor entered to intercept the previous satellite.

III. Results

The code was validated by using orbit geometries with known delta Vs. The obvious place to start validation is with two satellites in the same orbit with different phases. With a time-of-flight starting at 2 to 3 orbital periods, the delta Vs should be small since a phasing maneuver is all that is required for interception. The second validation step was to take two satellites with a known intercept time and see if the calculated delta V for that time was close to zero.

Once the code was validated, the next orbits looked at were satellites in two completely different orbits and then determining if the hypothesis that the lowest delta V will occur at the relative line of nodes was true or not. Next, some historical orbits were examined. Using a polar orbit with XSS-11-like characteristics, delta Vs for targets such as the International Space Station (ISS), an Israeli spy satellite, an Iridium satellite, and a Russian Earth imaging satellite (Resurs DK) were used. The full set of satellites considered is listed in Table III-1.

Table III-1. Orbits Used for Analysis

Name	Inclination (degrees)	Right Ascension of the Ascending Node (degrees)	Eccentricity	Argument of Perigee (degrees)	Mean Anomaly (degrees)	Mean Motion (revs/day)
Satellite 1 in Phasing Only	45.0242	0.0567	0.0007672	267.2596	272.7984	15.22470883
Satellite 2 in Phasing Only	45.0242	0.0567	0.0007704	266.9928	93.0659	15.22470101
Satellite 1 in Natural Intercept	23.021	0.0271	0.0004507	262.3204	98.2633	16.29163400

Satellite 2 in Natural Intercept	58.0235	0.0631	0.0009632	269.0869	92.0145	16.2702579
XSS-11	98.7964	0.1055	0.0010201	271.5869	88.4544	14.09867153
Hubble Space Telescope	28.4699	39.8730	0.0003468	87.3861	272.7128	15.00446074
ISS (Zarya)	51.6431	27.1693	0.0006592	183.9969	235.6151	15.72167165
OFEQ 5	143.4668	69.8412	0.0024973	170.9836	189.1353	15.05442684
Iridium 8	86.3995	164.5450	0.0002193	83.9303	276.2138	14.34216593
Resurs DK	69.9328	68.6438	0.0155888	57.4777	304.1371	15.35226411
MTI	97.2007	185.0417	0.0018749	295.1236	127.6792	15.11650443
ESSAIM-1	98.2341	329.4839	0.0003441	28.7950	331.3451	14.70161446

The set of satellites investigated is not exhaustive but they provide a wide range of inclinations along with a wide range of Right Ascensions of the Ascending Node (RAAN). All satellites examined were in nearly circular orbits (eccentricities close to zero). An extension of this work would be to run this analysis on satellites in Molniya orbits, although a suitable propagator would need to be found. Argument of perigee and mean motion were chosen to be somewhat random. Monte Carlo simulations would provide better randomization. Time constraints prevented research into this area. Mean motion is related to the more familiar semi-major axis "a" by

$$n = \sqrt{\frac{\mu}{a^3}} \tag{4}$$

where n is the mean motion, “a” is the length of the orbit’s semi-major axis, and μ is the standard gravitational parameter (the product of the universal gravitational constant and the mass of the central body).

The satellite OFEQ 5 is interesting in that it was placed in a retrograde (inclination greater than 90°) orbit. Imaging satellites are placed into orbits that are close to being polar. A satellite in a nearly polar, but retrograde, orbit can take advantage of the sun synchronous properties of certain orbits. Although in most cases sun synchronous orbits are used for Earth remote sensing applications (to take advantage of the sun being at the same spot when the satellite passes over the location of interest on the Earth), it should be noted that the sun also illuminates objects above the surface (that is, in space). Depending upon the sensors and how they are used, having certain sun angles may be required for properly gathering data.

Using Analytical Graphic Inc’s Satellite Toolkit, the following images were constructed for illustrative purposes. The orbit of the interceptor is shown in Figure III-1, the orbit of the Hubble Space Telescope (HST) is shown in Figure III-2, the orbit of the ISS is shown in Figure III-3 and OFEQ-5 is shown in Figure III-4. Orbital mechanics is a 4 dimensional problem (time being taken into account) and so the following figures are used to help the reader properly understand the geometries involved.

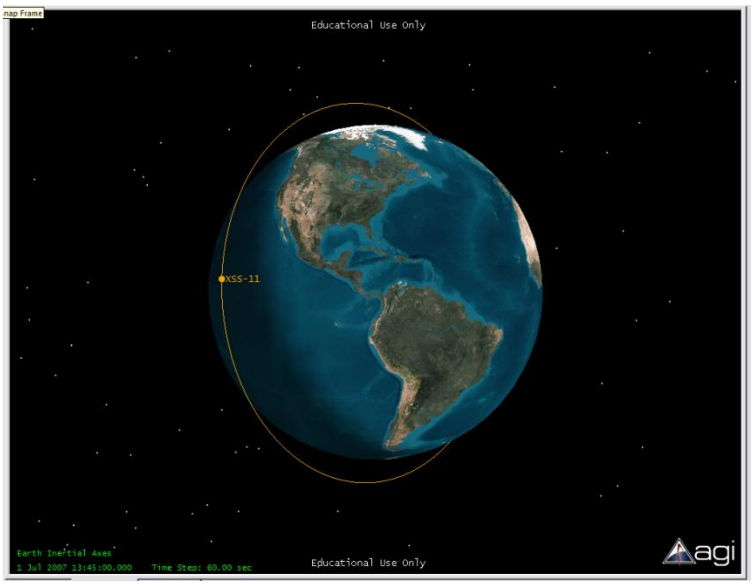


Figure III-1. Orbit of XSS-11

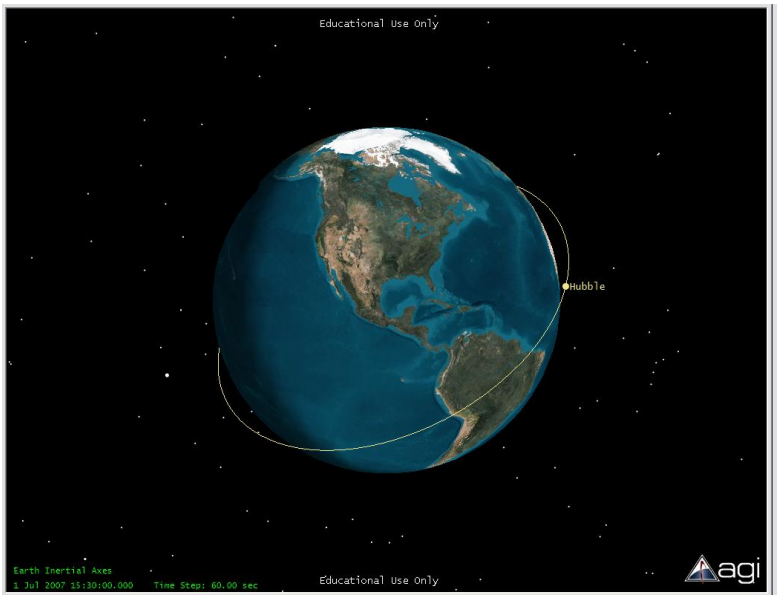


Figure III-2. Orbit of Hubble Space Telescope

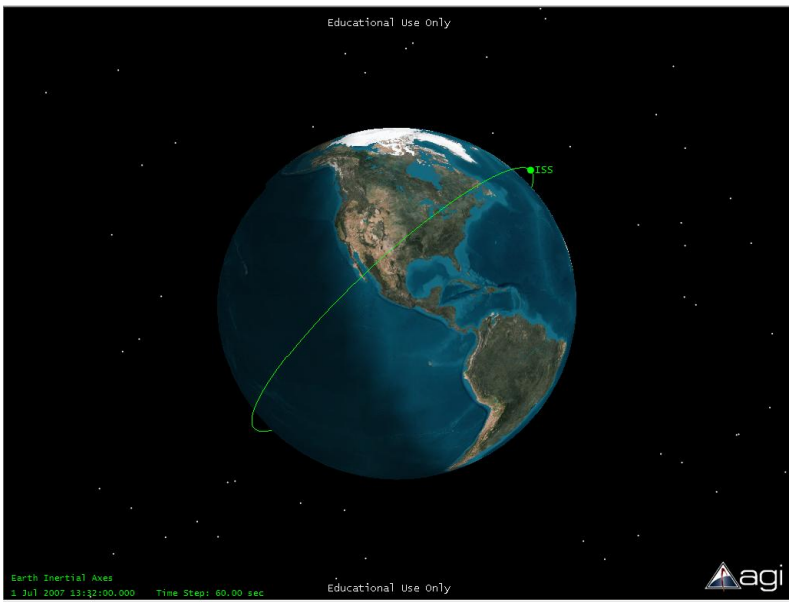


Figure III-3. Orbit of International Space Station

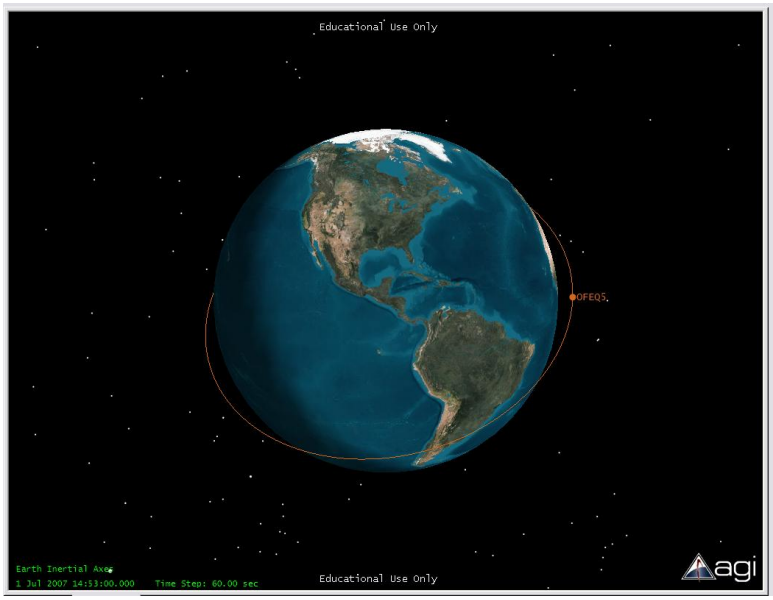


Figure III-4. Orbit of OFEQ 5

The SGP4 propagator requires the use of TLEs. TLEs contain the six elements that completely define an orbit but also include "solve-for" parameters. As Vallado points out, "The solve-for parameters allow us to estimate dynamical and measurement model parameters like the drag coefficient and measurement biases" (Vallado D. , Fundamentals of Astrodynamics and Applications, 2001, p. 703). One such term is called "bstar" which is the measure of the drag coefficient. The first and second time derivatives of the mean motion are other solve-for parameters. These are used to better understand how the spacecraft is accelerating, the biggest stochastic factor being the atmosphere. In this research for multiple intercepts, the 6 COEs were updated after each burn, but the bstar, first and second derivative terms were reused in the updated TLE. The updated TLE string for the interceptor was then used in the next loop. A higher fidelity model would need to recalculate the first and second time derivates for the new orbit, but not the bstar term as the physical shape of the interceptor would not change. Atmospheric drag acts on a satellite according to Equation 5

$$\vec{F} = -\frac{1}{2}\rho v^2 A C_d \hat{v} \tag{5}$$

where F is the force of the drag, ρ is the density of the fluid, v^2 is the speed of the object, squared, relative to the fluid, A is the reference area, C_d is the drag coefficient, $\hat{v}$ is the unit vector indicating the direction of the velocity. (The negative sign showing the drag force is opposite to that of the velocity.)

Knowledge of the orbit of a satellite is never perfect and is always changing (due to perturbations) and so TLEs need to be periodically refreshed. A rule of thumb for TLEs is to update them every 2 weeks for satellites in LEO and 4-5 weeks for higher orbits.

All of the collected data used the option in the code for the computer to pick the smallest delta V burn. For all scenarios, a burn window of 100 minutes (with a 1 minute time step), an intercept window of 100 minutes (with 1 minute time step) and a separation of windows of 1, 7, 30 and 90 days were used. For the single burn case, the data found was recorded in Table III-2.

Table III-2. Single Burn Data

Orbit Matchup	Burn Cost (km/sec)	Time Elapsed (days)
180° Phasing Difference	0.0014358	89.99
Different Inclinations	0.0068081	29.96
XSS to HST	0.2141090	8.17
XSS to ISS	0.1735544	0.98
XSS to OFEQ 5	0.1516724	2.06
XSS to Iridium 8	0.0389150	3.55
XSS to Resurs DK	0.1791356	2.06
XSS to MTI	0.1346047	2.13
XSS to ESSAIM-1	0.1045169	8.22

The first row has the lowest burn cost which lines up with the fact that it came from the scenario of two satellites only separated by phase. The time elapsed (time from burn to intercept) for this phasing maneuver is close to 90 days which makes sense since a phasing maneuver gets cheaper the longer the time-of-flight is, as shown for circular orbits in Equation 6.

$$\Delta V = \sqrt{\frac{\mu}{a_1}}\left(\sqrt{2-\left(1-\frac{\sigma_0}{2k\pi}\right)^{-\frac{2}{3}}}-1\right) \tag{6}$$

where a_1 is the semi-major axis of the circular orbit, σ_0 is the phase difference of the two satellites, and k is an arbitrary positive integer. Equation 7 shows the period of the phasing orbit and Equation 8 shows the relationship with k. (All equations from (Titus, 2008))

$$T_2 = 2\pi\sqrt{\frac{a_2^3}{\mu}} \tag{7}$$

$$a_2 = a_1\left(1-\frac{\sigma_0}{2k\pi}\right)^{\frac{2}{3}} \tag{8}$$

The variables T_2 and a_2 are the period and semi-major axis of the phasing orbit. The variable k is the number of revolutions you will travel in your phasing orbit before the interception. Examining Equations 6 through 8 reveals a classic problem in engineering. The faster the phasing is done, the more delta V is required. Conversely, the longer the phasing takes, the lower its required delta V becomes.

The different inclinations matchup is a scenario where two satellites have the same altitude but differ by inclination. Here the lowest delta V found is about five times that of the phasing only scenario. This low delta V occurred with an elapsed time of about 30 days. The delta V here was expected to be low since this is another type of phasing problem. Both spacecraft have the same altitude and so for them to intercept at the relative line of nodes, only their orbital periods need to be changed. The time elapsed was not close to 90 days though as was the previous case. This occurred because the orbits of the 2 satellites were designed to come close to interception on their own within 100 minutes of their epochs. This was done by hand in STK and so their position vectors never matched entirely at 100 minutes and so the code found a time in the future where they completely match with a small burn. The rest of the rows are taken from the real world. Average burn cost was 0.142358 km/sec, around which all of the burns seem to be clustered (with the exception of the XSS to Iridium burn). The average burn to intercept time was 3.88 days. Here the numbers appear to be grouped in three clusters. XSS to HST and ESSAIM-1 took about 8 days, XSS to ISS took about a day, and the rest took around 2 days.

Many delta V plots were created while running the code. Not all of them will be used here as common characteristics can be seen in a few of them. On all delta V plots, the vertical axis is the burn window while the horizontal axis is the intercept window. Both axes are always 100 minutes long with 1 minute increments. The MATLAB color-bar displays the delta V values in Canonical Units. Figure III-5 to Figure III-9 come from the phasing only scenario with the 1, 7, 30, and 90 days of separation between the burn and intercept windows. The arrays from which the plots are drawn are created by using the horizontal axis as the intercept (or target) window and the vertical axis as the burn (or interceptor)

window. Each point then represents the delta V for that combination. The colors represent the magnitude for the delta V at each point. The colors go from red (highest value) to violet (lowest value). In all figures shown violet represents values around 0 km/sec and red represents values around 16 km/sec. The white, or empty squares are MATLAB coded Not-a-Number (NaN) values. These occur whenever the code returns no solutions, negative time-of-flight or, more frequently, at collinear positions. In the first figure the white squares are diagonal at 45° since both satellites are in the same orbit.

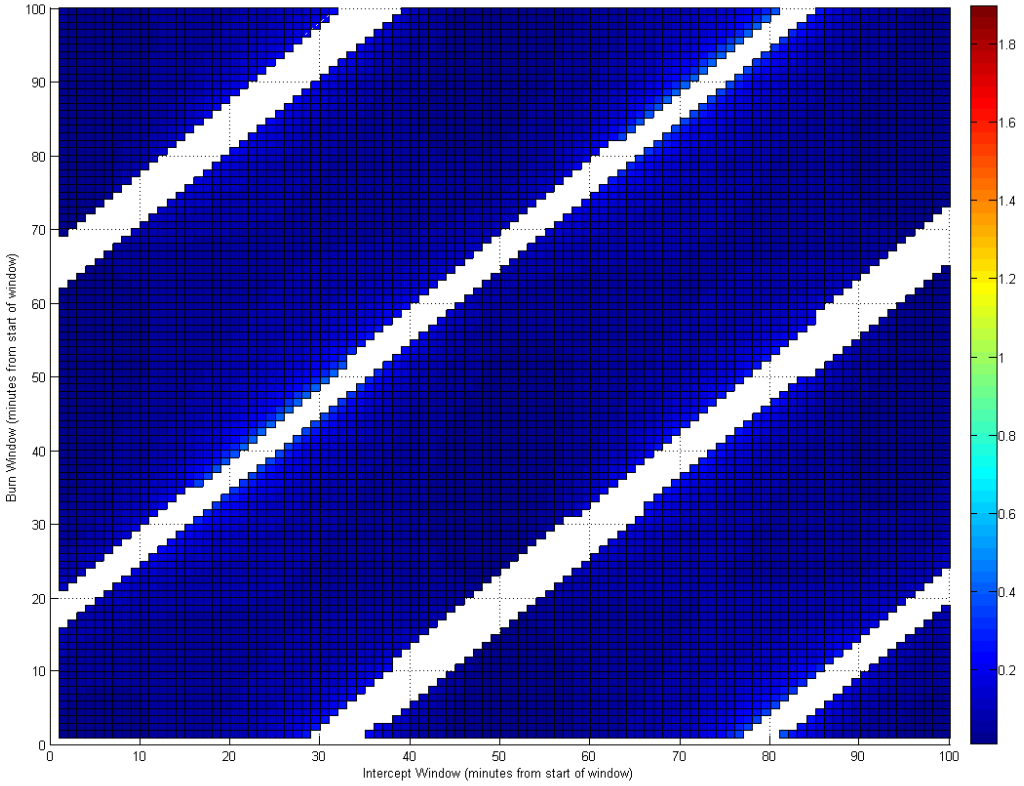


Figure III-5. Phasing Only Intercept Solutions

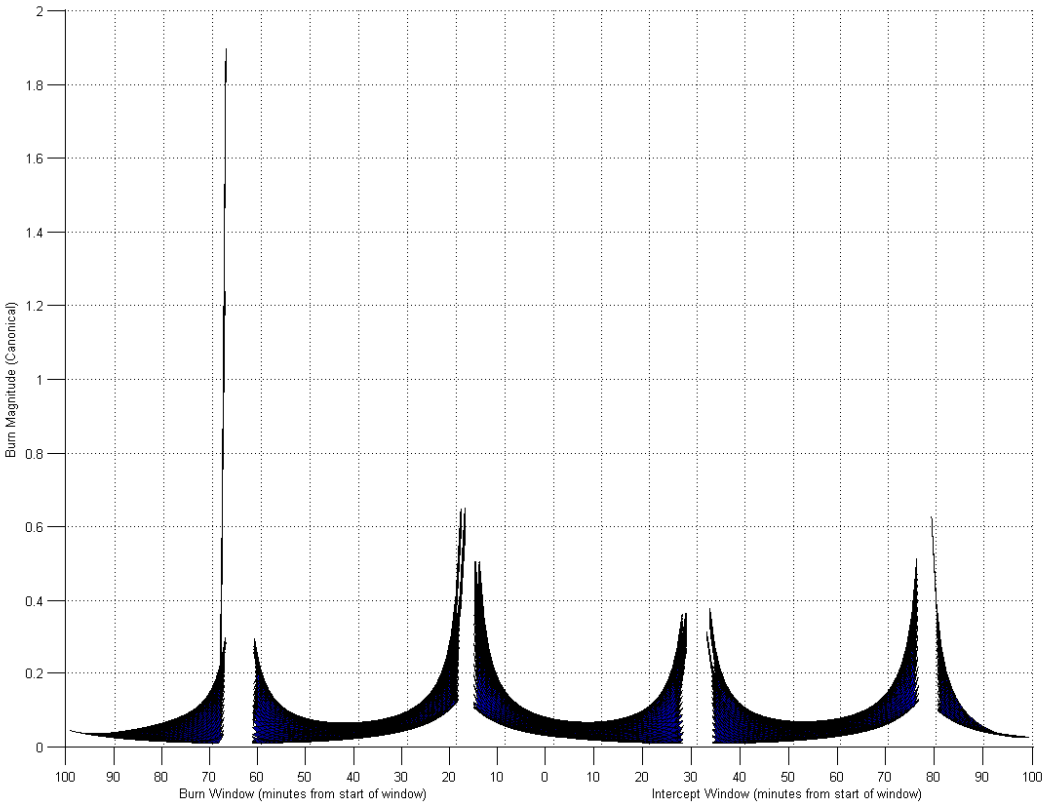


Figure III-6. Side-on View of Phasing Only Intercept Solution

The dark blues and violets in Figure III-5 indicate that most of the delta Vs being shown are low valued as expected. Figure III-6 is a side on view of Figure III-5. The numbers in Figure III-6 are presented in Canonical Units, where 1 represents 7.9 km/sec. As can be seen, as the collinear case approaches the cost increases. The high values seen on the left are quite noticeable and are probably artifacts of the p-iteration method used.

Figure III-7 shows the plot drawn at 7 days. The satellites gradually drift out of their original orbits as orbital perturbations slowly work on them. The oblateness of the Earth is the main cause of orbital perturbations. The 45° stripe of white squares are little by little turning into red squares and the remaining white squares are grouping together. As the squares take on different delta V values, the

lowest values begin to emerge on vertical lines going through the white squares. Vertical lines represent a fixed target location with changing burn locations. The vertical violet stripes go through the group of white squares, which show they are at the line of nodes. Their low values then seem to prove that intercepting at the line of nodes is indeed the cheapest option. Horizontal lines represent a constant burn point with changing interception points.

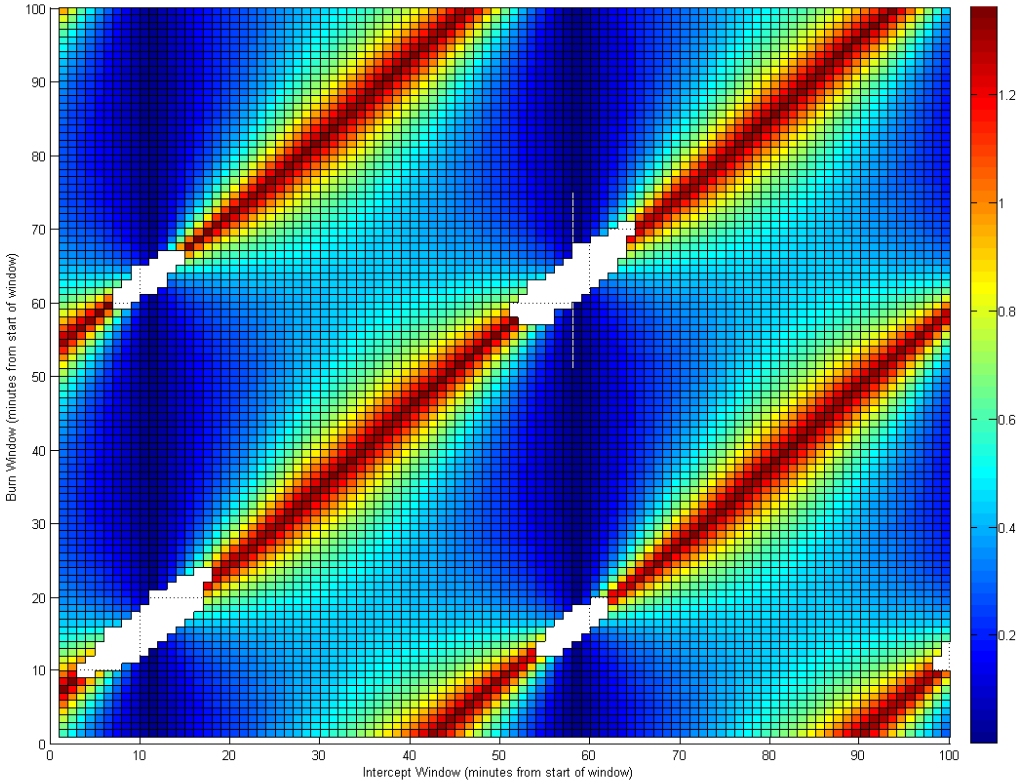


Figure III-7. 7 Day Perturbation of Phasing Only

Figure III-8 shows the 30 day case. Both satellites are now clearly in different orbits due to orbital perturbations. The periodic vertical line shows the line of nodes. Following a horizontal line results in a periodic delta V value, as the interception point moves around the orbit of the target. This occurs because the required interception orbit requires an inclination change that goes from zero (when

the target is at the relative line of nodes) up to 90°. Finally, Figure III-9 shows that the orbits have changed so much that it looks like the cases taken from actual satellites. Figures III-10 and III-11 show the values found for an intercept of the ISS at 1 and 30 days of burn/intercept window separation. Finally, Figure III-12 and Figure III-13 show the windows for the XSS to Resurs scenario at 1 and 90 days respectively. The 90 day figure is noticeable in that it looks like both satellites are in the same orbit with only a phase difference.

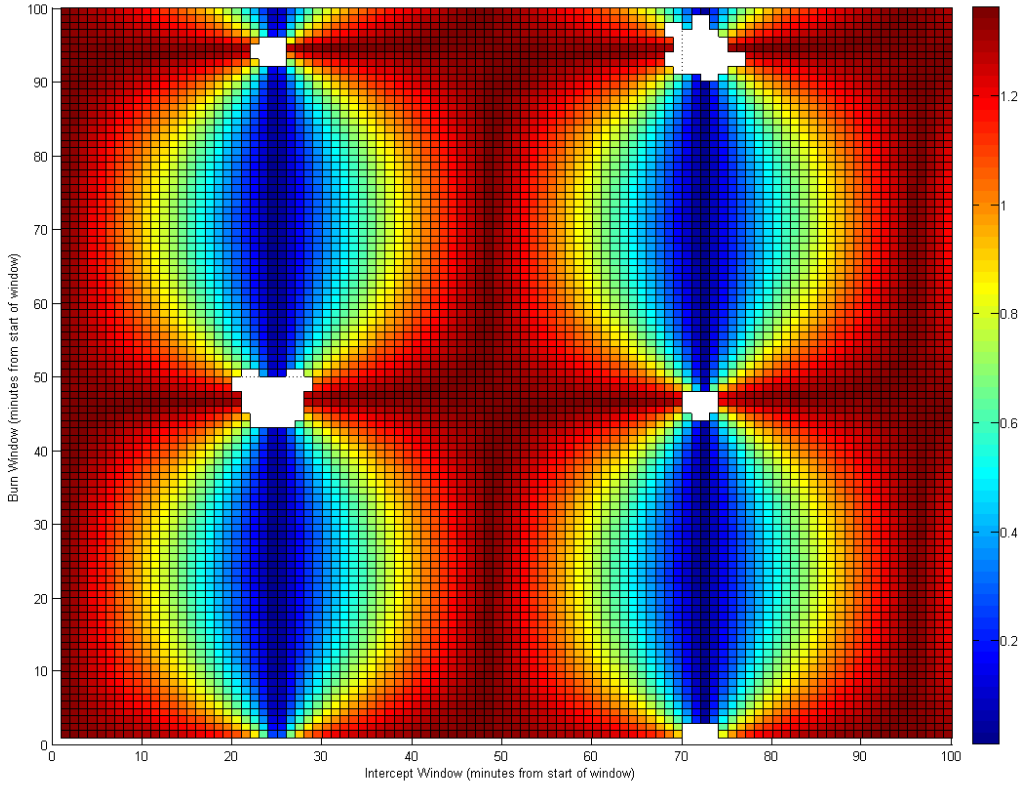


Figure III-8. 30 Day Perturbation of Phasing Only

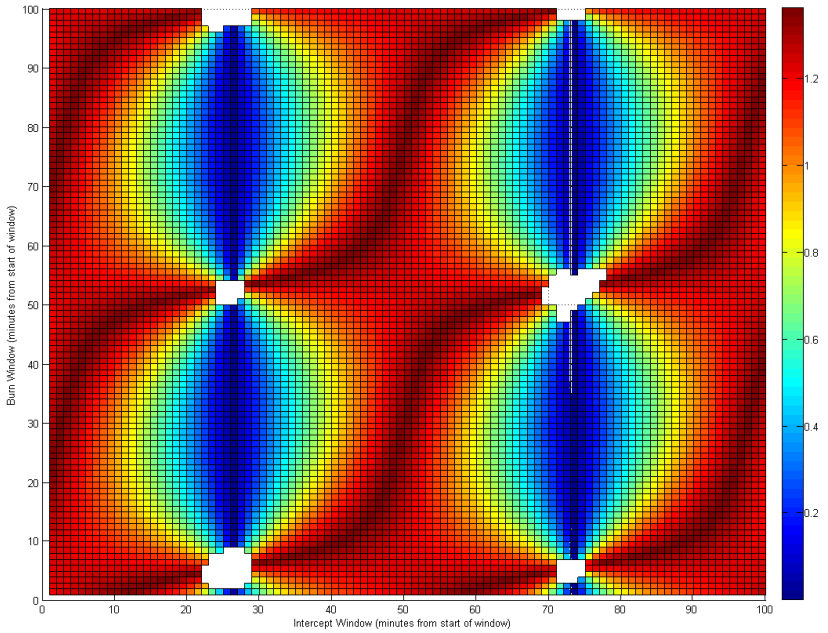


Figure III-9. 90 Day Perturbation of Phasing Only

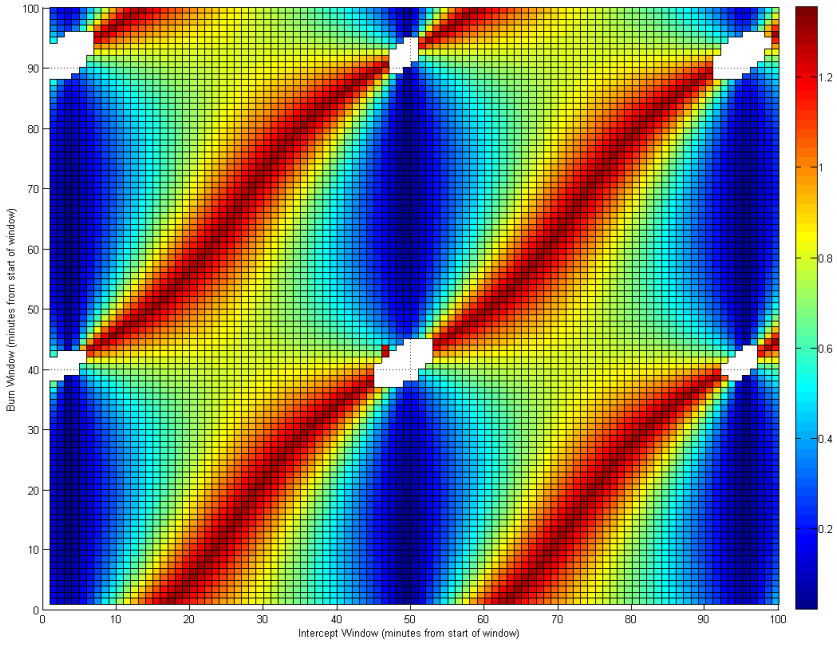


Figure III-10. XSS to ISS with 1 Day of Separation

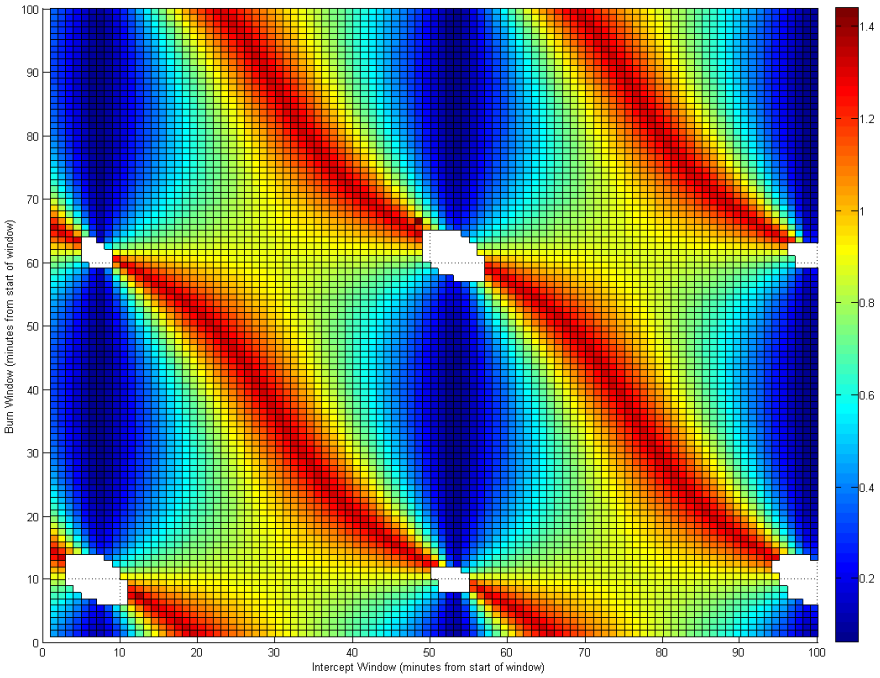


Figure III-11. XSS to ISS with 30 Days of Separation

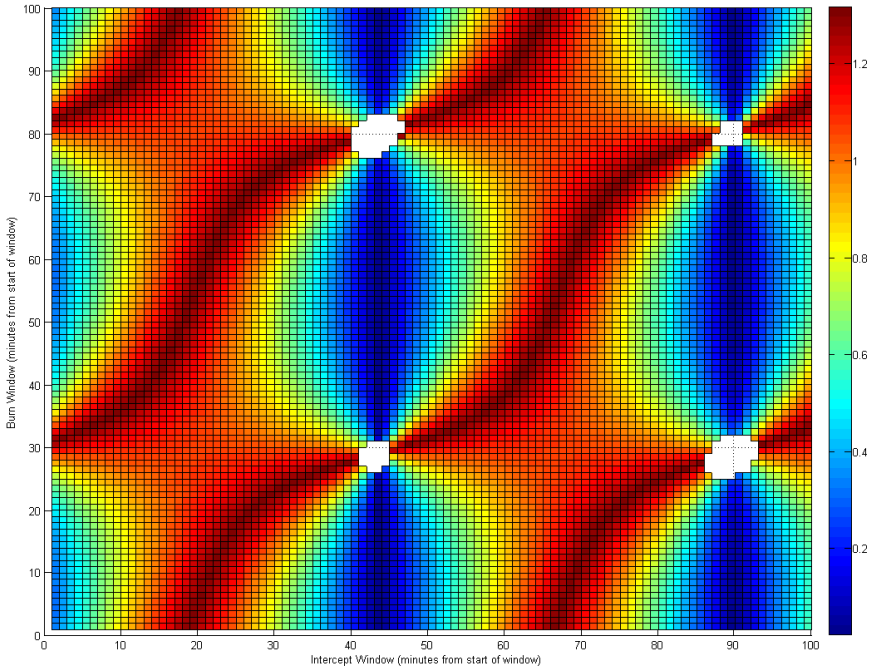


Figure III-12. XSS to Resurs with 1 Day of Separation

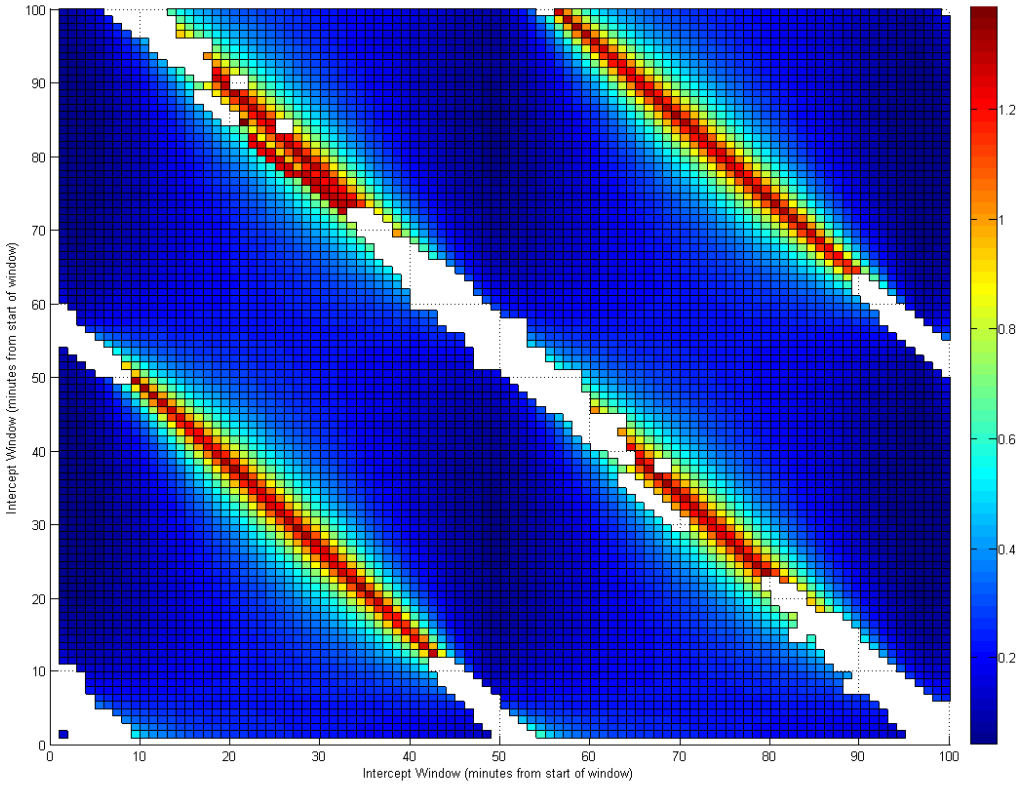


Figure III-13. XSS to Resurs at 90 Days of Separation

Table III-3. Multiple Burn Data

Orbit Matchup	Burn Cost (km/sec)	Burn to Intercept (days)
XSS to HST then ISS then ISS toolbag	0.491911541	17.14
XSS to OFEQ 5 then Resurs DK	0.17025	32.11
XSS to MTI then ESSAIM-1	0.16141181	32.16
XSS to Resurs DK then OFEQ 5	0.238678879	92.06
XSS to ESSAIM-1 then MTI	0.180495	15.27

Table III-3 shows the data collected for the multiple burn scenarios. It is interesting to note that the delta V cost for the multiple burn cases is only slightly higher than the single burn case. The time cost though is higher by about an order of magnitude. Table III-4 shows the breakdown of the intervening

orbits. The "burn wait time" column records where in the 100 minute burn window the burn occurred at.

Table III-4. Further Analysis on Multiple Burn

Orbit Intersecting	Mean Motion of intersecting orbit (revs per day)	Target Mean Motion	Burn Cost (km/sec)	Burn to Intercept Time (days)	Burn Wait Time (minutes)
XSS	14.09867153	-	-	-	-
XSS to HST	14.40722009	15.00446074	0.2141	8.169	84
HST to ISS	14.97076587	15.72167165	0.17487	1.629	66
ISS to toolbag	15.31197669	15.75798010	0.1029358	7.34	16
XSS to OFEQ	14.89605353	15.05442684	0.151672396	2.06	38
OFEQ to Resurs DK	14.98394887	15.35226411	0.018577615	30.05	17
XSS to MTI	14.87737695	15.11650443	0.1346	2.13	24
MTI to ESSAIM	14.79886034	14.70161446	0.026805523	30.03	56
XSS to Resurs DK	14.91235549	15.35226411	0.179135645	2.06	10
Resurs to OFEQ	14.95883894	15.05442684	0.059541653	90.00	79
XSS to ESSAIM	14.40091297	14.70161446	0.104516874	8.22	42
ESSAIM to MTI	14.85365980	15.11650443	0.075980866	7.05	94

Burn costs look to be grouped around 0.113 km/sec (the average). Burn to intercept times appear to be grouped into 1 to 2 days, 7 to 8 days and 30 to 90 days for the satellites examined.

Figure III-14 plots the wait times for each burn in all of the multi-burn cases. The interceptor is given a 100 minute window to make the burn in. If the window is too constrained we would see the times bunching up on one end of the window or the other. If the burns are made in a uniformly distributed manner they would not be correlated to each other. By observation, we can see that there appears to be no pattern to the wait times. In addition, the average burn wait time is 42.82 minutes. We would expect an average of 50 minutes if the distribution was centered about the middle of the window.

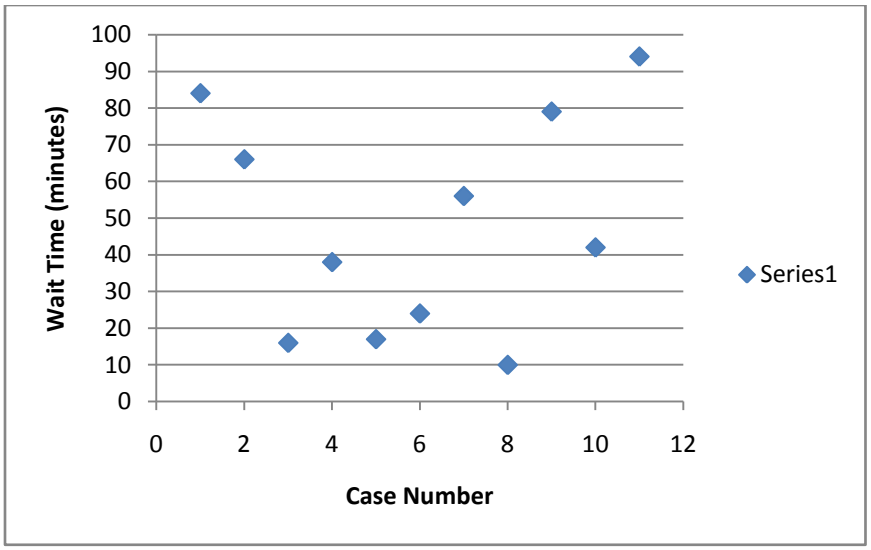


Figure III-14. Burn Wait Time Analysis

Figure III-15 shows the cost of each burn plotted against the elapsed time (from burn to intercept) for each burn. The data seems to be broken up into two groups. Burns greater than 0.075 resulted in elapsed times of fewer than 10 days. Burns less than that take 30 to 90 days. These low cost burns then seem to be changing the interceptor's period whereas the larger burns are changing the geometry of the orbit. Examining the change in mean motion will tell if this geometric change is occurring by a change of the semi-major axis.

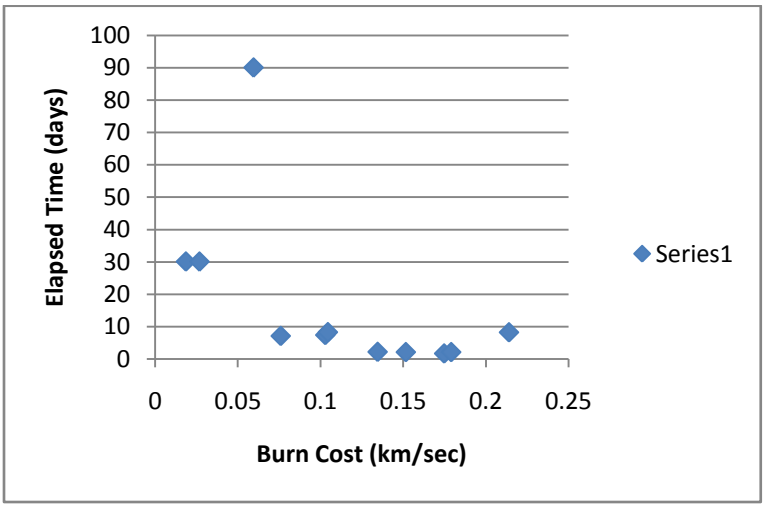


Figure III-15. Elapsed Time vs Burn Cost

Figure III-16 plots the change in mean motion of the interceptor versus the cost of that burn. A linear trendline was than drawn through the data points. The R^2 value here is interesting in that it is almost 0.5 (0.4949). This tells us that there is some correlation between the two variables.

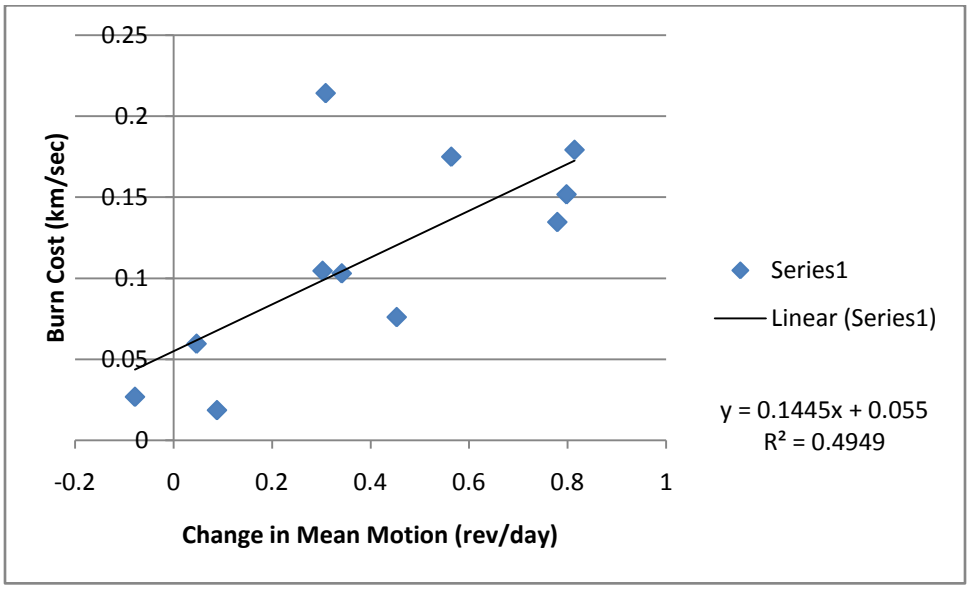


Figure III-16. Trending Burn Cost vs Change in Mean Motion

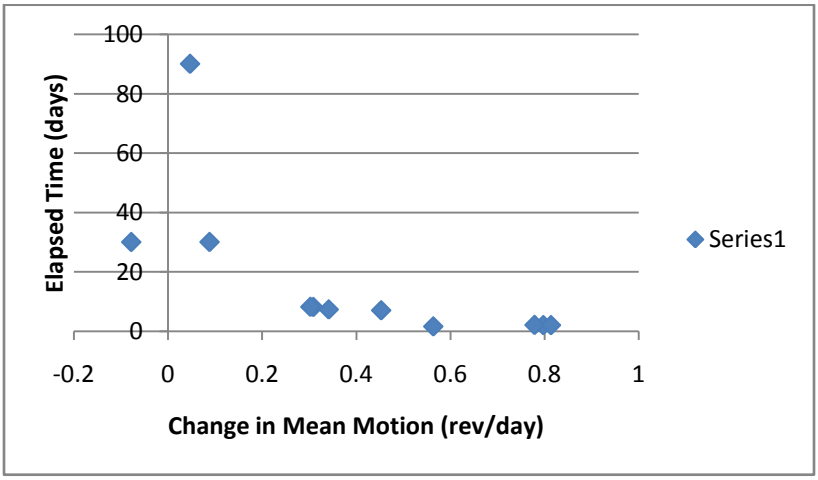


Figure III-17. Elapsed Time vs Change in Mean Motion

Figure III-17 does the same as above except it plots the change in mean motion against the elapsed time. This presents similar data as Figure III-15.

Ideally the whole solution space would be searched. That is, only one 100 minute window is looked at for 1, 7, 30 and 90 day periods. Is this window selection representative of the entire solution space? If not, what would be a better window to use? Orbit periods of the orbits under review are typically lower than 100 minutes. The delta V figures show at least two groups of white squares per vertical and horizontal line, which shows that at least one full orbit is completed within the search window.

IV. Further Research

There are many areas for additional study. A much greater data set should be collected, constructed by doing runs generated from random burn and intercept windows from the XSS-11 like orbit to the orbits of interest. Using randomized starting positions for the different orbits would allow the generic case to be studied. Examining burn and intercept windows several orbital periods long would increase the number of relative line of nodes crossing and thus more low valued delta V's could be found. Finding the lowest delta V path from these runs would create a firmer understanding of the necessary average wait time for the cheapest cost course. The exact tradeoff between delta V and time is not well understood. How much time could be gained if instead of picking the lowest delta V burn, the fifth lowest was used instead? Knowing this function (fuel savings vs time savings) would greatly enhance the mission planning process.

Integrating a solution for the repeated intercept problem would also greatly improve the mission planning capability of the developed software. Chioma and Titus did some work in this area (Chioma & Titus, 2008); it would be interesting to see how this could be used in the mission planning process.

Chemical engines are not the only type of power plant available to the spacecraft designer. Electric engines have been used with great success in operational satellites. These low thrust engines provide superior efficiency over their chemical brethren, which makes them attractive to spacecraft developed for a long operational lifetime. As already mentioned, the tradeoff in using such engines is the low thrust they provide. The software developed was for spacecraft that could instantaneously change orbits. Electric thrusters, on the other hand, slowly push a satellite into different orbits, in a spiral fashion. Thus, a different algorithm would need to be developed for the intercept problem. The solution would invariably have a longer elapsed time then the solutions found here, so any time critical missions would have to be accomplished by another spacecraft.

An extension of this work would be to run this analysis on satellites in Molniya orbits as well as deep space orbits, although a suitable propagator would need to be found. The forces exerted on a satellite vary depending upon the altitude the satellite is at. Thus a satellite in geostationary orbit would require a propagator that modeled forces that have lower effect on satellites in low Earth orbit.

V. Bibliography

Air Force Research Labratory. (2005, February). *Space Vehicles Fact Sheets.* Retrieved January 21, 2009, from Kirtland Air Force Base: http://www.kirtland.af.mil/afrl_vs/factsheets/index.asp

Air Force Research Labratory. (2005, December). *Space Vehicles Fact Sheets.* Retrieved January 21, 2009, from Kirtland Air Force Base: http://www.kirtland.af.mil/afrl_vs/factsheets/index.asp

Bate, R. R., Mueller, D. D., & White, J. E. (1971). *Fundamentals of Astrodynamics.* New York: Dover Publications Inc.

Chioma, V. J. (2007). *Navigation Solutions to Enable Space Superiority through the Repeated Intercept Mission.* WPAFB: AFIT.

Chioma, V. J., & Titus, N. A. (2008). Navigation Soultions for the Repeated-Intercept Mission with Constrained Maneuver Time. *Journal of Spacecraft and Rockets* , 116-121.

Hoots, F. R., & Roehrich, R. L. (1980). *Spacetrack Report No. 3.*

Titus, N. (2008, January). MECH 532 Lecture Notes. *AFIT*. WPAFB, OH.

Vallado, D. A., Crawford, P., Hujsak, R., & Kelso, T. S. (2006). Revisiting Spacetrack Report #3. 88.

Vallado, D. (2009, January 14). *Celestrak: Astrodynamics Software*. Retrieved January 21, 2009, from CelesTrak Web site: celestrak.com/software/vallado-sw.asp

Vallado, D. (2001). *Fundamentals of Astrodynamics and Applications.* Boston: Kluwer Academic Publishers.

Wiesel, W. E. (1997). *Spaceflight Dynamics.* New York: McGraw-Hill.

Appendix A. TLEs used

Listed here are the TLEs used in the MATLAB script.

Satellite 1 where satellites are only separated by phase

1 99992U 07182.50000000 -.00000812 00000-0 -34300-4 0 00008
2 99992 045.0242 000.0567 0007672 267.2596 272.7984 15.22470883000011

Satellite 2 where satellites are only separated by phase

1 99993U 07182.50000000 -.00000198 00000-0 -83624-5 0 00009
2 99993 045.0242 000.0567 0007704 266.9928 093.0659 15.22470101000018

Satellite 1 where satellites will intercept at 12:44 on 1 July

1 99994U 07182.50000000 -.00009755 00000-0 -21362-5 0 00008
2 99994 023.0210 000.0271 0004507 262.3204 098.2633 16.29163400000014

Satellite 2 where satellites will intercept at 12:44 on 1 July

1 99995U 07182.50000000 .00007120 00000-0 19968-5 0 00009
2 99995 058.0235 000.0631 0009632 269.0869 092.0145 16.27025790000011

XSS-11 like interceptor
1 99991U 09011.00000000 -.00000650 00000-0 -36969-3 0 00000
2 99991 098.7964 000.1055 0010201 271.5869 088.4544 14.09867153000011

HST
1 20580U 90037B 09012.20862878 .00000287 00000-0 93588-5 0 3031
2 20580 28.4699 39.8730 0003468 87.3861 272.7128 15.00446074826409

ISS (ZARYA)
1 25544U 98067A 09011.85056079 .00010407 00000-0 81779-4 0 495
2 25544 51.6431 27.1693 0006592 183.9969 235.6151 15.72167165581364

ISS DEB [TOOLBAG]
1 33442U 98067BL 09011.25286709 .00032848 00000-0 21202-3 0 846
2 33442 51.6442 29.2806 0003042 156.4997 203.6516 15.75798010 8270

Iridium 8

1 24792U 97020A 09013.51466258 -.00000006 00000-0 -90977-5 0 4699
2 24792 86.3995 164.5450 0002193 83.9303 276.2138 14.34216593612211

RESURS-DK 1
1 29228U 06021A 09012.03705442 .00000925 00000-0 21723-4 0 9107

2 29228 69.9328 68.6438 0155888 57.4777 304.1371 15.35226411144401

OFEQ 5
1 27434U 02025A   09012.07822485 .00000718 00000-0 17397-4 0 8252
2 27434 143.4668 69.8412 0024973 170.9836 189.1353 15.05442684369752

MTI
1 26102U 00014A   09012.12933296 .00000597 00000-0 38532-4 0 2330
2 26102 97.2007 185.0417 0018749 295.1236 127.6792 15.11650443485822

ESSAIM-1
1 28494U 04049C   09012.18614689 .00000013 00000-0 10668-4 0 6209
2 28494 98.2341 329.4839 0003441 28.7950 331.3451 14.70161446218169

The two line element set is defined as (from http://celestrak.com/columns/v04n03/)

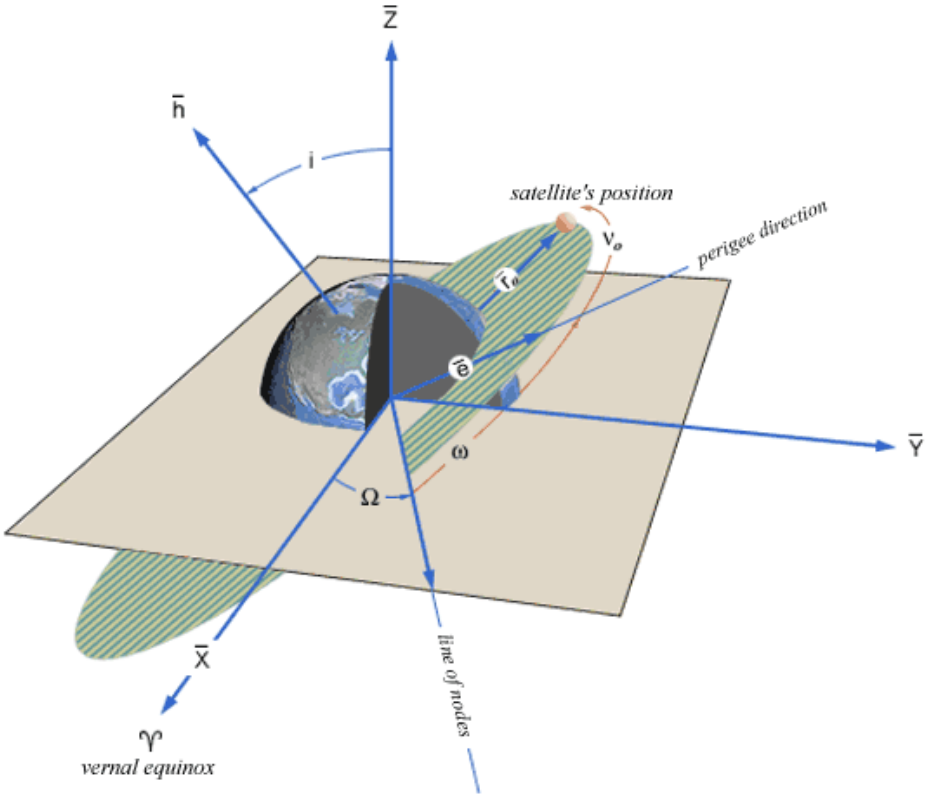
Line 1

Field	*Column*	*Description*
1.1	01	Line Number of Element Data
1.2	03-07	Satellite Number
1.3	08	Classification
1.4	10-11	International Designator (Last two digits of launch year)
1.5	12-14	International Designator (Launch number of the year)
1.6	15-17	International Designator (Piece of the launch)
1.7	19-20	Epoch Year (Last two digits of year)
1.8	21-32	Epoch (Day of the year and fractional portion of the day)
1.9	34-43	First Time Derivative of the Mean Motion
1.10	45-52	Second Time Derivative of Mean Motion (decimal point assumed)
1.11	54-61	BSTAR drag term (decimal point assumed)
1.12	63	Ephemeris type
1.13	65-68	Element number
1.14	69	Checksum (Modulo 10) (Letters, blanks, periods, plus signs = 0; minus signs = 1)

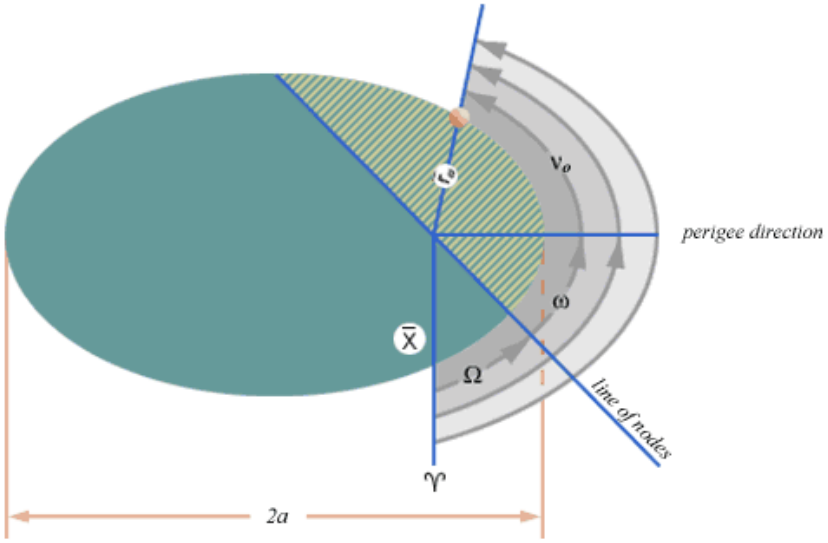
Line2

Field	*Column*	*Description*
2.1	01	Line Number of Element Data
2.2	03-07	Satellite Number
2.3	09-16	Inclination [Degrees]
2.4	18-25	Right Ascension of the Ascending Node [Degrees]
2.5	27-33	Eccentricity (decimal point assumed)
2.6	35-42	Argument of Perigee [Degrees]
2.7	44-51	Mean Anomaly [Degrees]
2.8	53-63	Mean Motion [Revs per day]
2.9	64-68	Revolution number at epoch [Revs]
2.10	69	Checksum (Modulo 10)

Appendix B. Classical Orbital Elements



a - defines the size of the orbit
e - defines the shape of the orbit
i - defines the orientation of the orbit with respect to the Earth's equator.
ω - defines where the low point, perigee, of the orbit is with respect to the Earth's surface.
Ω - defines the location of the ascending and descending orbit locations with respect to the Earth's equatorial plane.
ν - defines where the satellite is within the orbit with respect to perigee.



From (http://spaceflight.nasa.gov/realdata/elements/graphs.html)

Appendix C. Source Code

Below is the source code for “missionplanning.m”, “piter.m” and “tofp.m”

```
close all;clear all;clc

% This software was written by Capt Barry Witt in Dec 2008.  In general, it
% displays graphically the computed delta V's for an interceptor and a target
% using TLE's for the satellites and a user defined search window.  The
% user defines one window for the interceptor (when the burn will take
% place) and one to many windows for the target (when the interception will
% take place).  For each burn/intercept combination a delta V is calculated
% into an array, and the array is displayed as a surface plot.  Further,
% multiple targets can be looked at in succession.  Please note that for
% the new interceptor orbit, only the 6 COEs and epoch time are updated.

% The software can be divided up into 4 parts.  The first part loads the
% TLEs into the SGP4 propagator.  The second part creates the burn and
% intercept windows.  The third part calculates and displays the delta Vs.
% The fourth part finds the intercept time and classical orbit elements of
% the new orbit and then updates the interceptor TLE with the new epoch
% time and COEs.  It then loops it over again.

% Feature To-Do list
% 1. Add the option for repeated intercepts
% 2. Simplify initialization of SGP4

% Bug To-Do List
% 1.



disp(' ')
disp('Welcome to the Mission Planning Tool for Satellites')
disp('Please verify TLE data has been properly inserted into this m-file')
numtgts=input('How many targets will be looked at?  ');

total_time=0;  %this variable keep tracks of the total time since original
epoch of the interceptor.
total_burn=0;  %this variable keep tracks of the total total delta V.



whichconst=84;typerun='m';typeinput='d';%These are variables for twoline2rv.m
Using '84 earth model, with a manual input


DU2Km=6378.135;
TU2min=13.44686457;
VU2KmSec=7.90536828;
Re=1+200/6378;  %this is the radius of the earth plus a safety factor of 200
km.
```

```
%------------------------Place Interceptor TLE Below-----------------------
%using longstr1 = first line and longstr2 = second line


%XSS-11
longstr1='1 99991U          09011.00000000 -.00000650  00000-0 -36969-3 0
00000';
longstr2='2 99991 098.7964 000.1055 0010201 271.5869 088.4544
14.09867153000011';


%Sat 1 180 off phase
% longstr1='1 99992U          07182.50000000 -.00000812  00000-0 -34300-4 0
00008';
% longstr2='2 99992 045.0242 000.0567 0007672 267.2596 272.7984
15.22470883000011';

%Sat1 different inclinations
% longstr1='1 99994U          07182.50000000 -.00009755  00000-0 -21362-5 0
00008';
% longstr2='2 99994 023.0210 000.0271 0004507 262.3204 098.2633
16.29163400000014';



disp(' ')
disp('The propogator will now initialize.')
disp('Please select dates that will cover the entire time period under
consideration.')
disp(' ')


disp('For the interceptor')
[satrec(1,1), startmfe, stopmfe, deltamin] = twoline2rv(whichconst, longstr1,
longstr2, typerun,typeinput); %Interceptor at satrec(1,1)




%------------------------Place Target(s) TLE Data Below--------------------
% Use naming convetion for TLEs as longstrx and longstrx+1 where x is 2
% times the target number plus 1.


%Target 1 TLE Data
```

```
longstr3='1 28494U 04049C   09012.18614689  .00000013  00000-0  10668-4 0
6209';
longstr4='2 28494  98.2341 329.4839 0003441  28.7950 331.3451
14.70161446218169';


disp(' ')
disp('For Target 1')
[satrec(1,2), startmfe, stopmfe, deltamin] = twoline2rv(whichconst, longstr3,
longstr4, typerun,typeinput); %Target 1 at satrec(1,2)


%Target 2 TLE Data


longstr5='1 26102U 00014A   09012.12933296  .00000597  00000-0  38532-4 0
2330';
longstr6='2 26102  97.2007 185.0417 0018749 295.1236 127.6792
15.11650443485822';




disp(' ')
disp('For Target 2')
[satrec(1,3), startmfe, stopmfe, deltamin] = twoline2rv(whichconst, longstr5,
longstr6, typerun,typeinput); %Target 2 at satrec(1,3)


 %Target 3 TLE Data

% longstr7='1 33442U 98067BL  09011.25286709  .00032848  00000-0  21202-3 0
846';
% longstr8='2 33442  51.6442  29.2806 0003042 156.4997 203.6516 15.75798010
8270';
%
%
% disp(' ')
% disp('For Target 3')
% [satrec(1,4), startmfe, stopmfe, deltamin] = twoline2rv(whichconst,
longstr7, longstr8, typerun,typeinput); %Target 3 at satrec(1,4)









 %---------------------------End of Target TLE Data-----------------------
```

```
 for tgtctr=1:numtgts  %this is the begenning of the loop for each target




disp(' ')  %this creates the burn window
disp('Please select dates and times that cooincide with the information
entered above.')
disp(' ')
disp('For the interceptor, for the burn window.')
i_ep=input(['How long after day ',num2str(satrec(1,1).epochdays),' in year
',num2str(satrec(1,1).epochyr),'  should the window begin? (in minutes)  ']);
%FIX this to reflect the time of intercept from the previous burn
i_win=input('How long should the window be? (in minutes)  ');
i_tstep=input('What time step should be used within the window? (in minutes)
');

i_blocks=floor(i_win/i_tstep);  %this divides the window up into blocks, from
which to calculate the delta Vs.




for ctr=1:i_blocks+1
    tsince(ctr)=i_ep+(ctr-1)*i_tstep;
    [satrec(1,1), r_xss(:,ctr), v_xss(:,ctr)] =
sgp4(satrec(1,1),tsince(ctr));  %generate r and v for the interceptor
end


disp(' ')
disp(['For target ',num2str(tgtctr),', for the intercept window.'])
nwin=input('How many intercept windows?  ');  %this allows multuple intercept
windows



for nctr=1:nwin  %this is the begenning of the creation loop for intercept
windows

disp(' ')
disp(['For intercept window ',num2str(nctr),'.'])
t_ep(nctr)=input(['How long after day
',num2str(satrec(1,tgtctr+1).epochdays),' in year
',num2str(satrec(1,tgtctr+1).epochyr),' should the window begin? (in minutes)
']);
t_win(nctr)=input('How long should the window be? (in minutes)  ');
```

```
t_tstep(nctr)=input('What time step should be used within the window? (in
minutes)  '); %this is similar to creating the burn window above.

t_blocks(nctr)=floor(t_win(nctr)/t_tstep(nctr));




for ctr2= 1:t_blocks(nctr)+1
    tsince2(ctr2,nctr)=t_ep(nctr)+(ctr2-1)*t_tstep(nctr);
    [satrec(1,tgtctr+1), r_hst(:,ctr2,nctr),v_hst(:,ctr2,nctr)] =
sgp4(satrec(1,tgtctr+1),tsince2(ctr2,nctr));  %generate r and v for the
target
end


deltaep=365-satrec(1,1).epochdays+((satrec(1,tgtctr+1).epochyr-
satrec(1,1).epochyr)-1)*365+satrec(1,tgtctr+1).epochdays;  %this is the
number of days between the TLE epochs of the interceptor and the target



for ctr3=1:i_blocks+1
    for ctr4=1:t_blocks(nctr)+1
        tdays(ctr3,ctr4,nctr)=deltaep+tsince2(ctr4,nctr)/(60*24)-
tsince(ctr3)/(60*24);  %Time of flight in days between each time step in the
burn and intercept windows.
    end
end

end


r_xss=r_xss/DU2Km; %Piter works only with Canonical Units
v_xss=v_xss/VU2KmSec;
r_hst=r_hst/DU2Km;
v_hst=v_hst/VU2KmSec;



i_energy=0.5*(norm(v_xss(:,1)))^2-1/norm(r_xss(:,1));
i_a=-1/(2*i_energy);
period=(2*pi*sqrt(i_a^3))*TU2min;  % this is the period of the interceptor in
minutes/orbit of the pre-burn orbit.


nrev=round(tdays/(period/(24*60)));  %nrev is the number of revolutions of
the proposed interception orbit for piter.  This is the number of revolutions
the interceptor would take if it's orbital period did not change.


% the code below calculates the delta Vs for all of the burn and intercept
% combinations.
```

```
for nctr=1:nwin  %this is the begenning of the loop for each intercept window



for ctr5=1:i_blocks+1  %this counter advances the interceptor
    for ctr6=1:t_blocks(nctr)+1  %this advances the target
        if tdays(ctr5,ctr6,nctr) < 0
            mdelv(ctr5,ctr6,nctr)=NaN;  %if the windows overlap, then some of
the combinations will have a negative tof
        else
            for ctrc=1:5  %this is the nrev counter
                if nrev(ctr5,ctr6,nctr)-3+ctrc < 0  %if the nrev that is
proposed is less then zero, it needs to be skipped
                    mdelv3(ctrc)=NaN;
                    mdelv5(ctrc)=NaN;
                    continue
                end

[nsoln,vr_xss,v21]=piter(r_xss(:,ctr5),r_hst(:,ctr6,nctr),tdays(ctr5,ctr6,nct
r),0,nrev(ctr5,ctr6,nctr)-3+ctrc);

[nsoln2,vr_xss2,v22]=piter(r_xss(:,ctr5),r_hst(:,ctr6,nctr),tdays(ctr5,ctr6,n
ctr),1,nrev(ctr5,ctr6,nctr)-3+ctrc);  %piter needs to be run in short way and
long way modes
                if nsoln < eps   %this code checks to see if piter returned
an actual vector or not
                    mdelv3(ctrc)=NaN;
                end
                if nsoln2 < eps
                    mdelv5(ctrc)=NaN;
                end
                if nsoln < eps && nsoln2 < eps  %if both piter runs contain
no solutions, continue to the next nrev
                    continue
                end
                for stu=1:nsoln
                    energy=0.5*(norm(vr_xss(:,stu)))^2-
1/norm((r_xss(:,ctr5)));
                    a=-1/(2*energy);
                    H=cross(r_xss(:,ctr5),vr_xss(:,stu));
                    e=cross(vr_xss(:,stu),H)-
r_xss(:,ctr5)/norm(r_xss(:,ctr5));
                    rp=a*(1-e);
                    if rp-Re < 0   %this codes checks to see if the perigee
height is above earth or not
                        vr_xss(:,stu)=[NaN;NaN;NaN];
                    end
                    delv2(:,stu)=vr_xss(:,stu)-v_xss(:,ctr5);
                    mdelv2(stu)=norm(delv2(:,stu));
                end
```

```
                if nsoln > eps  %this prevents mdelv3 from being re-defined
if nsoln was 0.
                    mdelv3(ctrc)=min(mdelv2);
                end

                for stu=1:nsoln2
                    energy=0.5*(norm(vr_xss2(:,stu)))^2-
1/norm((r_xss(:,ctr5)));
                    a=-1/(2*energy);
                    H=cross(r_xss(:,ctr5),vr_xss2(:,stu));
                    e=cross(vr_xss2(:,stu),H)-
r_xss(:,ctr5)/norm(r_xss(:,ctr5));
                    rp=a*(1-e);
                    if rp-Re < 0   %this codes checks to see if the perigee
height is above earth or not
                        vr_xss2(:,stu)=[NaN;NaN;NaN];
                    end
                    delv4(:,stu)=vr_xss2(:,stu)-v_xss(:,ctr5);
                    mdelv4(stu)=norm(delv4(:,stu));
                end
                if nsoln2 > eps %this prevents mdelv5 from being re-defined
if nsoln2 was 0.
                    mdelv5(ctrc)=min(mdelv4);
                end
            end
            mdelv(ctr5,ctr6,nctr)=min(min(mdelv3,mdelv5));  %this picks the
smallest delta V out of all of the nrev runs.
        end
    end
end



figure(nctr);surf(mdelv(:,:,nctr));title(['Plot for window ',num2str(nctr),',
target ',num2str(tgtctr),'.']);colorbar   %a new figure is created for each
intercept window.
xlabel({['Position of the target using a window of ',num2str(t_win(nctr)),'
minutes with a time step of ',num2str(t_tstep(nctr)),' minutes.'];['The
beginning of the window was ',num2str(t_ep(nctr)),' minutes after day
',num2str(satrec(1,tgtctr+1).epochdays),' in year
',num2str(satrec(1,tgtctr+1).epochyr),'.']});
ylabel({['Position of the interceptor using a window of ',num2str(i_win),'
minutes with a time step of ',num2str(i_tstep),' minutes.'];['The beginning
of the window was ',num2str(i_ep),' minutes after day
',num2str(satrec(1,1).epochdays),' in year
',num2str(satrec(1,1).epochyr),'.']});
zlabel('Burn Magnitude (Canonical)');

end


%The code below allows to either have MATLAB pick the smallest burn
%value, or have the user input the desired burn.
```

```
disp(' ')
disp('After reviewing the plots, please indicate what mode to use to pick the
burn/intercept combination.')
xdecide=input('User determines burn (1) or MATLAB picks first smallest Delta
V (2)  ');

if xdecide < 2

% this is for the user specified burn.

disp(' ')
disp('Using the data cursor')
disp('Pick the requested burn')
npick=input('From what window?  ');
bcol=input('What is the x value?  ');  %this is the target value
brow=input('What is the y value?  ');  %this is the interceptor value

disp(' ')
disp(['The burn picked was ',num2str(mdelv(brow,bcol,npick)),' DU/TU'])
%this code allows the user to verify his pick
disp(['The burn occured at ',num2str(tsince(brow)),' minutes after day
',num2str(satrec(1,1).epochdays),' in year
',num2str(satrec(1,1).epochyr),'.'])
disp(['The intercept occrred at ',num2str(tsince2(bcol,npick)),' minutes
after day ',num2str(satrec(1,tgtctr+1).epochdays),' in year
',num2str(satrec(1,tgtctr+1).epochyr),'.'])

disp(' ')
total_time=total_time+tsince2(bcol,npick)+deltaep*1440;
total_burn=total_burn+mdelv(brow,bcol,npick);
disp(['The total time since original epoch is ',num2str(total_time),'
minutes.'])
disp(['The total burn is ',num2str(total_burn),' '])

for ctrc=1:5  %as all of the velocity required vectors were not saved from
above, piter must be re-run for the selected burn/intercept combination
    if nrev(brow,bcol,npick)-3+ctrc < 0
        continue
    end

[nsoln,vra_xss(:,:,ctrc),v21]=piter(r_xss(:,brow),r_hst(:,bcol,npick),tdays(b
row,bcol,npick),0,nrev(brow,bcol,npick)-3+ctrc);

[nsoln2,vrb_xss(:,:,ctrc),v22]=piter(r_xss(:,brow),r_hst(:,bcol,npick),tdays(
brow,bcol,npick),1,nrev(brow,bcol,npick)-3+ctrc);
    if nsoln<eps && nsoln2<eps
        continue
    end
    for stu=1:nsoln
        energy=0.5*(norm(vra_xss(:,stu,ctrc)))^2-1/norm((r_xss(:,brow)));
        a=-1/(2*energy);
        H=cross(r_xss(:,brow),vra_xss(:,stu,ctrc));
        e=cross(vra_xss(:,stu,ctrc),H)-r_xss(:,brow)/norm(r_xss(:,brow));
        rp=a*(1-e);
```

```
        if rp-Re < 0   %this codes checks to see if the perigee hieight is
above earth or not
            vra_xss(:,stu,ctrc)=[NaN;NaN;NaN];
        end
    end
    for stu=1:nsoln2
        energy=0.5*(norm(vrb_xss(:,stu)))^2-1/norm((r_xss(:,brow)));
        a=-1/(2*energy);
        H=cross(r_xss(:,brow),vrb_xss(:,stu));
        e=cross(vrb_xss(:,stu),H)-r_xss(:,brow)/norm(r_xss(:,brow));
        rp=a*(1-e);
        if rp-Re < 0   %this codes checks to see if the perigee hieight is
above earth or not
            vrb_xss(:,stu,ctrc)=[NaN;NaN;NaN];
        end
    end
end




for ctrc=1:size(vra_xss,3)
    for ctrb=1:size(vra_xss(:,:,ctrc),2)
        if norm(vra_xss(:,ctrb,ctrc)) < eps   %if the code above skips a nrev
and then does a run later, MATLAB will put zeros in the skipped column
            mdelva(ctrb,ctrc)=NaN;
            continue
        end
        mdelva(ctrb,ctrc)=norm(vra_xss(:,ctrb,ctrc)-v_xss(:,brow));
    end
end



for ctrc=1:size(vrb_xss,3)
    for ctrb=1:size(vrb_xss(:,:,ctrc),2)
        if norm(vrb_xss(:,ctrb,ctrc)) < eps
            mdelvb(ctrb,ctrc)=NaN;
            continue
        end
        mdelvb(ctrb,ctrc)=norm(vrb_xss(:,ctrb,ctrc)-v_xss(:,brow));
    end
end



% Decide either a or b

deca=min(min(mdelva));  %this is used in determining which piter run has the
smalles delta V.
decb=min(min(mdelvb));

if deca<decb || isnan(decb)
```

```
    if size(mdelva,1)<2  %in some cases mdelva will only have one row, and so
a slightly different way of finding the correct v is needed
        [xc,xi]=min(mdelva);
        ctrcnum=xi;
        ctrbnum=1;
    else
        [xc,xi]=min(mdelva);
        [xxc,xxi]=min(xc);
        ctrbnum=xi(xxi);
        ctrcnum=xxi;
    end
    %nv_xss=vra_xss(:,ctrbnum,ctrcnum);  %this is the new velocity vector of
the internceptor at burn
    sway=0;
else
    if size(mdelvb,1)<2
        [xc,xi]=min(mdelvb);
        ctrcnum=xi;
        ctrbnum=1;
    else
        [xc,xi]=min(mdelvb);
        [xxc,xxi]=min(xc);
        ctrbnum=xi(xxi);
        ctrcnum=xxi;
    end
    %nv_xss=vrb_xss(:,ctrbnum,ctrcnum);  %this is the new velocity vector of
the internceptor at burn
    sway=1;
end


[nsoln,v11,vai_xss]=piter(r_xss(:,brow),r_hst(:,bcol,npick),tdays(brow,bcol,n
pick),sway,nrev(brow,bcol,npick)-3+ctrcnum);
nv_xss=vai_xss(:,ctrbnum);  %this is the velocity of the interceptor at
intercept




else  %this is the option for MATLAB determining the first smallest burn
    for nctr=1:nwin
        smdv(nctr)=min(min(mdelv(:,:,nctr)));
    end
    [bc,bi]=min(smdv);
    disp(['The burn picked was ',num2str(bc),' DU/TU'])
    npick=bi;
    [bc,bi]=min(mdelv(:,:,npick));
    [bbc,bbi]=min(bc);
    brow=bi(bbi); %this is the row of mdelv
    bcol=bbi; %this is the coloumn of mdelv

    %the code below was copied from the option above

    for ctrc=1:5
        if nrev(brow,bcol,npick)-3+ctrc < 0
```

```
            continue
        end

[nsoln,vra_xss(:,:,ctrc),v21]=piter(r_xss(:,brow),r_hst(:,bcol,npick),tdays(b
row,bcol,npick),0,nrev(brow,bcol,npick)-3+ctrc);

[nsoln2,vrb_xss(:,:,ctrc),v22]=piter(r_xss(:,brow),r_hst(:,bcol,npick),tdays(
brow,bcol,npick),1,nrev(brow,bcol,npick)-3+ctrc);
    if nsoln<eps && nsoln2<eps
        continue
    end
    for stu=1:nsoln
        energy=0.5*(norm(vra_xss(:,stu,ctrc)))^2-1/norm((r_xss(:,brow)));
        a=-1/(2*energy);
        H=cross(r_xss(:,brow),vra_xss(:,stu,ctrc));
        e=cross(vra_xss(:,stu,ctrc),H)-r_xss(:,brow)/norm(r_xss(:,brow));
        rp=a*(1-e);
        if rp-Re < 0   %this codes checks to see if the perigee hieight is
above earth or not
            vra_xss(:,stu,ctrc)=[NaN;NaN;NaN];
        end
    end
    for stu=1:nsoln2
        energy=0.5*(norm(vrb_xss(:,stu)))^2-1/norm((r_xss(:,brow)));
        a=-1/(2*energy);
        H=cross(r_xss(:,brow),vrb_xss(:,stu));
        e=cross(vrb_xss(:,stu),H)-r_xss(:,brow)/norm(r_xss(:,brow));
        rp=a*(1-e);
        if rp-Re < 0   %this codes checks to see if the perigee hieight is
above earth or not
            vrb_xss(:,stu,ctrc)=[NaN;NaN;NaN];
        end
    end
    end




    for ctrc=1:size(vra_xss,3)
        for ctrb=1:size(vra_xss(:,:,ctrc),2)
            if norm(vra_xss(:,ctrb,ctrc)) < eps
                mdelva(ctrb,ctrc)=NaN;
                continue
            end
            mdelva(ctrb,ctrc)=norm(vra_xss(:,ctrb,ctrc)-v_xss(:,brow));
        end
    end




    for ctrc=1:size(vrb_xss,3)
        for ctrb=1:size(vrb_xss(:,:,ctrc),2)
            if norm(vrb_xss(:,ctrb,ctrc)) < eps
```

```
                mdelvb(ctrb,ctrc)=NaN;
                continue
            end
            mdelvb(ctrb,ctrc)=norm(vrb_xss(:,ctrb,ctrc)-v_xss(:,brow));
        end
    end

% Decide either a or b

deca=min(min(mdelva));
decb=min(min(mdelvb));



if deca<decb || isnan(decb)
    if size(mdelva,1)<2
        [xc,xi]=min(mdelva);
        ctrcnum=xi;
        ctrbnum=1;
    else
        [xc,xi]=min(mdelva);
        [xxc,xxi]=min(xc);
        ctrbnum=xi(xxi);
        ctrcnum=xxi;
    end
    %nv_xss=vra_xss(:,ctrbnum,ctrcnum);  %this is the new velocity vector of
the internceptor at burn
    sway=0;

else
    if size(mdelvb,1)<2
        [xc,xi]=min(mdelvb);
        ctrcnum=xi;
        ctrbnum=1;
    else
        [xc,xi]=min(mdelvb);
        [xxc,xxi]=min(xc);
        ctrbnum=xi(xxi);
        ctrcnum=xxi;
    end
    %nv_xss=vrb_xss(:,ctrbnum,ctrcnum);  %this is the new velocity vector of
the internceptor at burn
    sway=1;
end

[nsoln,v11,vai_xss]=piter(r_xss(:,brow),r_hst(:,bcol,npick),tdays(brow,bcol,n
pick),sway,nrev(brow,bcol,npick)-3+ctrcnum);
nv_xss=vai_xss(:,ctrbnum);  %this is the velocity of the interceptor at
intercept

    disp(['The burn occured at ',num2str(tsince(brow)),' minutes after day
',num2str(satrec(1,1).epochdays),' in year
',num2str(satrec(1,1).epochyr),'.'])
```

```
    disp(['The intercept occrred at ',num2str(tsince2(bcol,npick)),' minutes
after day ',num2str(satrec(1,tgtctr+1).epochdays),' in year
',num2str(satrec(1,tgtctr+1).epochyr),'.'])

    disp(' ')
    total_time=total_time+tsince2(bcol,npick)+deltaep*1440;
    total_burn=total_burn+mdelv(brow,bcol,npick);
    disp(['The total time since original epoch is ',num2str(total_time),'
minutes.'])
    disp(['The total burn is ',num2str(total_burn),' '])

end



%nr_xss=r_xss(:,brow);    %this is the position vector of the interceptor at
burn.
nr_xss=r_hst(:,bcol,npick);  %this is the position vector of the interceptor
at interception

tint=tsince2(bcol,npick)+deltaep*1440;  %this is the time in minutes since
interceptor TLE epoch at intercept.

tburn=tsince(brow);  %this is the time in minutes since interceptor TLE epoch
at burn.

nr_xss=nr_xss*DU2Km; %converting units from Canonical to km/sec
nv_xss=nv_xss*VU2KmSec;


% I need to generate a new TLE based upon the r and v of the interceptor at
% intercept.  This problem is not easily done.  So I will create a TLE
% changing r and v into COEs and reusing the bstar and 1st/2nd derivative
% information from the original TLE.

[p,a,ecc,incl,omega,argp,nu,m,arglat,truelon,lonper ] = rv2coe
(nr_xss,nv_xss,398600.44);  %the 6 COES associated with r and v.
incl=rad2deg(incl);omega=rad2deg(omega);argp=rad2deg(argp);m=rad2deg(m);
%TLEs use degrees

T=2*pi*sqrt(a^3/398600.44);
mm=86400/T;  %mean motion in revs per day for the TLE

edays=tint/(24*60);

if edays+satrec(1,1).epochdays > 365  %this code determines the new epoch
year and epoch day
    xyears=floor((edays+satrec(1,1).epochdays)/365);
    newyear=satrec(1,1).epochyr+xyears;
    newday=edays+satrec(1,1).epochdays-(xyears*365);
else
    newyear=satrec(1,1).epochyr;
    newday=edays+satrec(1,1).epochdays;
end
```

```
eyear=num2str(newyear,'%02.0f');  %this updates the year and day in the
interceptor TLE
eday=num2str(newday,'%012.8f');
longstr1(19:20)=eyear;
longstr1(21:32)=eday;

ninc=num2str(incl,'%08.4f');    %this updates the COEs in the TLE
nraan=num2str(omega,'%08.4f');
necc=num2str(ecc,'%07.0f');
nargp=num2str(argp,'%08.4f');
nma=num2str(m,'%08.4f');
nmm=num2str(mm,'%011.8f');

longstr2(9:16)=ninc;
longstr2(18:25)=nraan;
longstr2(27:33)=necc;
longstr2(35:42)=nargp;
longstr2(44:51)=nma;
longstr2(53:63)=nmm;

disp(' ')
disp(['the new epoch year is ',eyear, '.'])
disp(['the new epoch day is ',eday,'.'])
disp(['The new mean motion is ',nmm,'.'])
disp(' ')

[satrec(1,1), startmfe, stopmfe, deltamin] = twoline2rv(whichconst, longstr1,
longstr2, typerun,typeinput);  %this changes satrec(1,1) to the new
interceptor orbit.


close all; clear smdv mdelv r_hst r_xss v_hst v_xss vra_xss vrb_xss   %this
clears out variables that are recreated in the next run, so old data is not
used.


 end   %this is the end statment for multuple targets
```

```
function [nsoln,v1,v2] = piter(r1,r2,tdays,iway,nrev)

% piter  Uses an iteration method to solve the two position vector and time
% of flight problem
%
% piter function has been translated from BASIC code written by Dr. William
Wiesel
% Translated by Captain Barry Witt August 2008
%
% [nsoln,v1,v2]=piter(r1,r2,tdays,iway,nrev)
%
% P iteration method, ref Bate, Mueller & White section 5.4
% ELLIPSES ONLY
% units are DU's, TU's, although input, t, is in days
% W. Wiesel, AFIT/ENY, November 1998
% BM&W referrs to Bate, Mueller & White, "Fundamental of Astrodynamics"
%
% NOTE:Piter does not work with colinear position vectors, despite a time
% difference between them.
%
% Outputs definitions
% nsoln - number of solutions found: 0 if it fails, 1 if it found 1
%         solution (usually < 1 rev case), or 2 solutions found (for > 1 rev
case)
% v1(i,isoln) - initial velocity vector, DU/TU, isoln = 1 or 2.  This is
% the velocity required at launch, or burn.   In the same frame as r1 and
% r2.
% v2(i,isoln) - final velocity vector, isoln = 1 or 2.  This is the
% velocity of the inteceptor at the interception point
%
% Inputs definitions
% r1 - the position vector of the interceptor at the initial time, in DU,
% in ECEF frame
% r2 - the position vector at the requested point of interception, in DU,
% in ECEF frame
% tdays is elapsed time of flight, in days
% iway = 0 for "short way", else for "long way"
% nrev = number of whole revolutions, 0 if <1.  Note: This is chosen
% independently of tof

xmu=1;

%calculate some preliminaries

nsoln=0;
TU2min=13.44686457;
%convert time of flight from days to TU

t = tdays * (1440 / TU2min);
rone=norm(r1);
rtwo=norm(r2);
cosdnu=dot(r1,r2)/(rone*rtwo);
```

```
%colinear case won't converge

if cosdnu>0.985
    %disp('Position vectors are colinear in the same direction. Piter
terminating.')
    nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN]; %i have chosen NaN as the
error return value
    return
elseif cosdnu<-0.995
    %disp('Position vectors are colinear in the opposite direction. Piter
terminating')
    nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
    return
end

%three auxillary quantities, BM&W 5.4-4

xk=rone*rtwo*(1-cosdnu);
xl=rone+rtwo;
xm=rone*rtwo*(1+cosdnu);

%parabolic limiting values for parameter p, BM&W 5.4-14, -15

pM=xk/(xl+sqrt(2*xm));
pp=xk/(xl-sqrt(2*xm));


%calculate discriminant for p limits, Denominator of BM&W 5.4-8

disc=4*xk*xk*xl*xl+4*xk*xk*(2*xm-xl*xl);  %The denominator in the book has a
p, but this does not

if disc<0  %no elliptic solutions
    %disp('There are no elliptic solutions.  Piter terminating')
    nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
    return
end


%BRANCH ON NUMBER OF EXPECTED SOLUTIONS

if nrev<eps
    % less than one rev case: this is the case explicitly covered in BM&W.
    % only one solution each way (short or long) is expected.

    % opening strategy: bisection, 10X, start just inside parabolic limits

    del=0.000001;
    plow=pM+del;
    phi=pp-del;


    [toflo,dtdp]=tofp(plow,rone,rtwo,cosdnu,xk,xl,xm,iway,nrev,0);
    if isnan(toflo)  %this code stops piter if tofp returns NaN for tof
```

```
        nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
        return
    end
    [tofhi,dtdp]=tofp(phi,rone,rtwo,cosdnu,xk,xl,xm,iway,nrev,0);
     if isnan(tofhi)
        nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
        return
    end

    for i=1:10  %bisect
        p=(plow+phi)/2;
        [tof,dtdp]=tofp(p,rone,rtwo,cosdnu,xk,xl,xm,iway,nrev,1);  %decide
which
        if isnan(tof)
            nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
            return
        end
        if (t-toflo)*(t-tof)<0  %low half brackets root
            phi=p;
            tofhi=tof;
        else  %root is in upper half
            plow=p;
            toflo=tof;
        end
    end

    %see if we are still pegged at one limit, and therefore never
    %converged....

    if (abs(p-pM-del)<0.0000000001) || (abs(p-pp+del)<0.0000000001)
        %disp('There was no convergence on the solution.  Piter
terminating.')
        nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];  %no convergence
        return
    end

    %endgame: Newton-Rhapson loop

    for i=1:10
        imode=1;
        [tof,dtdp]=tofp(p,rone,rtwo,cosdnu,xk,xl,xm,iway,nrev,imode);
        if isnan(tof)
            nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
            return
        end
        dt=tof-t;
        dp=-dt/dtdp;
        p=p+dp;
        if abs(dp)<0.00000000001
            break
        end
    end

    %didn't converge...check value of tof
    if (abs(t-tof))>0.00000001
```

```
        %disp('The difference between time requested and tof found was too
big, piter is stopping.')
        nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
        return
    end



    %final processing: get v1 and v2
    %calculate f & g functions, BM&W 5.2-3 thru 5.2-6

    f=1-rtwo*(1-cosdnu)/p;
    if iway<eps %short way
        sindnu=sqrt(1-cosdnu*cosdnu);
    else
        sindnu=-sqrt(1-cosdnu*cosdnu);
    end

    g=rone*rtwo*sindnu/sqrt(xmu*p);
    fdot=sqrt(xmu/p)*((1-cosdnu)/sindnu)*((1-cosdnu)/p-1/rone-1/rtwo);
    gdot=1-rone*(1-cosdnu)/p;

    for i=1:3
        v1(i,1)=(r2(i)-f*r1(i))/g;
        v2(i,1)=fdot*r1(i)+gdot*v1(i,1);
    end

    %declare success with one solution found
    nsoln=1;

else

    %Multirev case: two solutions expected.  This case is not explicitly
    %covered in BM&W.  The modifications for the multirev case are called
    %out in the code where they occur.  Mostly in the evaluation of time of
    %flight (tofp function).  Experimentation shows two solutions are
    %expected between the parabolic limits of pm and pp.

    %get initial approximations to both solutions by scanning p interval

    del=0.000001;
    plow=pM+del;
    phi=pp-del;

    pguess(1)=0;
    pguess(2)=0;
    tprev=0;
    nroots=0;
    for i=1:200
        p=plow*(200-i)/199+phi*(i-1)/199;
        [tof,dtdp]=tofp(p,rone,rtwo,cosdnu,xk,xl,xm,iway,nrev,0);
        if isnan(tof)
            nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
            return
```

```
        end
        if tprev<eps
            tprev=tof;
        else
            if (tof-t)*(tprev-t)<0  %we've bracked a root...which one?
                if pguess(1)<eps
                    pguess(1)=p;
                    nroots=1;
                else
                    pguess(2)=p;
                    nroots=2;
                    break
                end
            end
            tprev=tof;
        end
    end

    %both roots were not found

    if nroots<eps
        %disp('Unable to find both roots in piter.')
        nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
        return
    end


    %loop on solutions found.

    for ir=1:nroots  %endgame: Newton-Rhapson loop
        for i=1:10
            imode=1;

[tof,dtdp]=tofp(pguess(ir),rone,rtwo,cosdnu,xk,xl,xm,iway,nrev,imode);
            if isnan(tof)
                nsoln=0;v1=[NaN;NaN;NaN];v2=[NaN;NaN;NaN];
                return
            end
            dt=tof-t;
            dp=-dt/dtdp;

            pguess(ir)=pguess(ir)+dp;
            %check for negative p....abort
            if pguess(ir)<0
               break
            end
            if abs(dp)<0.0000000001
                break
            end
        end

        if pguess(ir)<0  %in Basic if this is true, the next ir loop goes
through
            nsoln=nsoln+1;   %if it skips an ir loop, the code neeeds nsoln
to be advanced
```

```
        else

            % final processing: get v1 and v2.  calculate f & g functions,
BM&W
            % 5.2-3 thru 5.2-6
            f=1-rtwo*(1-cosdnu)/pguess(ir);
            if iway<eps %short way
                sindnu=sqrt(1-cosdnu*cosdnu);
            else
                sindnu=-sqrt(1-cosdnu*cosdnu);
            end
            g=rone*rtwo*sindnu/sqrt(xmu*pguess(ir));
            fdot=sqrt(xmu/pguess(ir))*((1-cosdnu)/sindnu)*((1-
cosdnu)/pguess(ir)-1/rone-1/rtwo);
            gdot=1-rone*(1-cosdnu)/pguess(ir);
            nsoln=nsoln+1;
            for i=1:3
                v1(i,nsoln)=(r2(i)-f*r1(i))/g;
                v2(i,nsoln)=fdot*r1(i)+gdot*v1(i,ir);
            end

        end
    end

    %end <1 rev/multirev if block

end
```

```
function [tof,dtofdp] = tofp(p,rone,rtwo,cosdnu,xk,xl,xm,iway,nrev,imode)

%tofp function has been translated from BASIC code written by Dr. William
Wiesel.
%Translated by Captain Barry Witt August 2008.
%
%This calculates time of flight, tof, from supposed value p value for p
iteration
%method.  In addition this calculates dt/dp if imode ??? 0.
%Original code written by W. Wiesel, AFIT/ENY, Nov 1998.  "BM&W" refers to
%Bate, Mueller, and White, "Fundamentals of Astrodynamics", Dover.
%
%Outputs are tof,dtofdp. Inputs are
p,rone,rtwo,cosdnu,xk,xl,xm,iway,nrev,imode.
%
%Outputs definitions
%tof - time of flight in TU units
%dtofdp - rate of change of tof w.r.t. p
%
%Inputs definitions
%p - semi-latus rectum in ??? units
%rone - magnitude of the position vector of first satellite in ???? units
%rtwo - maginitude of the position vector of the second satellite in ???
units
%cosdnu - cosine of the angle between the two position vectors.
%xk - constant defined in BM&W
%xl - constant defined in BM&W
%xm - constant defined in BM&W
%iway - short/long way determination.  0 is short, else is long.
%nrev - number of revolutions.  0 is less then one case covered in BM&W.
%Note: this is chosen independently of tof
%Else is developed in the code.
%imode - if this is equal to zero then code only calculates tof.

xmu=1;

a=xm*xk*p/((2*xm-xl*xl)*p*p+2*xk*xl*p-xk*xk);  %calculate semimajor axis a,
BM&W 5.4-8

if a<0
    %disp('Accidental hyperbolic orbit')
    tof=NaN;
    dtofdp=NaN;
    return
end

%calculate f, fdot, g: BM&W 5.2-3, 5.2-5, 5.2-4

f=1-rtwo*(1-cosdnu)/p;
if iway<eps %short way
    sindnu=sqrt(1-cosdnu*cosdnu);
else
    sindnu=-sqrt(1-cosdnu*cosdnu);
```

```
end

if p<0
    %disp('p is negative.')
    tof=NaN;
    dtofdp=NaN;
    return
end

g=rone*rtwo*sindnu/sqrt(xmu*p);
fdot=sqrt(xmu/p)*((1-cosdnu)/sindnu)*((1-cosdnu)/p-1/rone-1/rtwo); %there's a
half angle trig sub here.

%change in eccentric anomaly, BM&W 5.4-9, 5.4-10

cosdE=1-rone*(1-f)/a;
sindE=-rone*rtwo*fdot/sqrt(xmu*a);
dE=acos(cosdE);

if sindE<0
    dE=2*pi-dE;
end

%ensure dE within 0-1 rev

if dE<0
    dE=dE+2*pi;
end

if dE>2*pi
    dE=dE-2*pi;
end

%add multiple revs to eccentric anomaly...required extension not in BM&W

dE=dE+2*pi*nrev;

%time of flight, BM&W 5.4-12

tof=g+sqrt(a*a*a/xmu)*(dE-sindE);

%is derivative not required?

 if imode<eps
     dtofdp=NaN;
     return
 end

 %calculate dt/dp

 %less-than-one rev derivative, BM&W 5.4-22
```

```
dtofdp=-g/(2*p)-1.5*a*(tof-sqrt(a*a*a)*2*pi*nrev-g)*((xk*xk+(2*xm-
xl*xl)*p*p)/(xm*xk*p*p))+sqrt(a*a*a/xmu)*2*xk*sindE/(p*(xk-xl*p));

%correction for multiple revs...nrev*d(DeltaE = 2 n pi)/da*da/dp

%NEEDS MU FIX

if nrev>0
    dtofdp=dtofdp-3*nrev*pi*sqrt(a*a*a*a*a)*((xk*xk+(2*xm-
xl*xl)*p*p))/(xm*xk*p*p);
end
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
26-03-2009	Master's Thesis	Jul 2008 - Mar 2009

4. TITLE AND SUBTITLE
Mission Planning for Close Proximity Satellites

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Witt, Barry, R, Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GA/ENY/09-M10

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Lab/Space Vehicles Directorate
Dr Thomas Lovell
3550 Aberdeen Ave SE, Bldg 593 Rm 10,
Kirtland AFB, NM 87117-5776
263-4132

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RVES

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Given an initial orbit and a set of other orbits of interest, the lowest change in velocity "path" between them is of high value. Software was developed to visualize the change in velocity required from a user defined window of burn to a user defined window of intercept. The time between burn and intercept is unrestricted. Multiple intercepts require searching an n! solution space, research then is focused on ways to pick optimal choices without fully calculating out the cost of doing all possible combinations. Some code was translated from BASIC and the rest was newly developed in MATLAB. The code was validated using orbits with known intercept solutions.

15. SUBJECT TERMS
mission planning, software, MATLAB, delta V, spacecraft, satellite

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			William Wiesel, AFIT/ENY
U	U	U	UU	70	19b. TELEPHONE NUMBER *(Include area code)* 785-3636 x4312